在线服务：视频库、源代码库、专业论坛、专家实时支持



Ajax+PHP程序设计实战详解

梁文新　宋强　刘凌霞　等编著



50段全程配音语音教学视频

全书实例源代码，使学习、分析、调试程序更方便

在线服务方式

在线服务网站：www.itzcn.com

QQ群在线服务：45368980、33925615、107423140

清华大学出版社

Ajax+PHP 程序设计实战详解

梁文新　宋强　等编著

清华大学出版社
北　京

内 容 简 介

本书介绍 Ajax 和 PHP 两种比较流行的技术，覆盖了 Ajax 和 PHP 技术的基本知识和应用场景。本书分为 5 篇，分别为：PHP 基础篇、PHP Web 应用篇、Ajax 基础开发篇、组合篇和实例篇。内容囊括了 PHP 开发的基础知识、Ajax 开发的基础知识、PHP+Ajax 组合开发 Web 2.0 程序和 PHP+Ajax 实例程序。本书最后介绍了一个教学视频网站后台管理系统，帮助读者全面掌握在实际项目中使用 Ajax 技术，提高对大型应用系统的整体把握，使读者熟练掌握 PHP+Ajax 技术。

本书适合网站开发人员、PHP 和 Ajax 开发人员以及 Web 开发爱好者学习和参考，也可以作为 PHP 和 Ajax 的教学参考书。

图书在版编目（CIP）数据

Ajax+PHP 程序设计实战详解 / 梁文新等编著. —北京：清华大学出版社，2010.7
ISBN 978-7-302-21295-9

Ⅰ. A…　Ⅱ. 梁…　Ⅲ. ①计算机网络 – 程序设计　②PHP 语言 – 程序设计　Ⅳ. TP393.09 TP312

中国版本图书馆 CIP 数据核字（2009）第 181939 号

责任编辑： 夏兆彦
责任校对： 徐俊伟
责任印制：
出版发行： 清华大学出版社　　**地　　址：** 北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 190×260　　**印　张：** 34　　**插　页：**　　**字　数：** 846 千字
版　　次： 2010 年 7 月第 1 版　　**印　次：** 2010 年 7 月第 1 次印刷
印　　数： 1～ 000
定　　价： 　元

产品编号：031594-01

FOREWORD

前言

PHP 是一种易于学习和使用的后台开发技术。用户只需要具备很少的编程知识，就可以使用 PHP 建立一个具有交互功能的 Web 站点。应用 PHP 有许多好处，如实用性强、功能强大、成本较低等。对于个人来说学习和使用 PHP 是一个很好的选择。

Ajax 技术为用户提供了更丰富的页面浏览体验，可以构建高度交互的 Web 应用。本书介绍了在 Web 2.0 的标准下结合 Ajax 技术进行各种 PHP Web 应用开发的方式和技巧。主要内容包括 CSS 和 XML 的基础、DOM 的应用、XMLHttpRequest 对象、JavaScript 开发工具、Ajax 客户端编程、与服务器通信以及将 Ajax 技术集成到 PHP 网站，还介绍了与 Ajax 相关的框架等技术。

1．本书内容

本书介绍了 Ajax 和 PHP 两种比较流行的技术，覆盖了 Ajax 和 PHP 技术基本知识和应用场景。本书分为 5 篇，分别为：PHP 基础篇、PHP Web 应用篇、Ajax 基础篇、PHP+Ajax 组合篇和综合实例篇，各篇主要内容如下。

第 1 篇：PHP 基础篇（第 1～4 章）。本篇首先向读者介绍 PHP 的发展历程，以及配置 PHP 环境，然后详细介绍 PHP 语法基础和面向对象特性，像常量、变量、数据类型、运算符、对象、类、属性、方法、继承、接口以及反射等。

第 2 篇：PHP Web 应用篇（第 5～8 章）。在本篇中首先介绍如何使用 PHP 提交与获取表单数据、登录信息；然后通过实例的方式详细阐述 cookie 和会话机制；最后介绍了文件 I/O 和数据库操作，以及在客户端如何处理 XML 文件等。

第 3 篇：Ajax 基础篇（第 9～14 章）。向读者介绍 Ajax 的概念、优势、运行机制以及第一个 Ajax 实例。然后，针对 Ajax 的核心组成部分进行详细介绍，包括 CSS、JavaScript、XML、XMLHttpRequest 及 DOM。

第 4 篇：PHP+Ajax 组合篇（第 15～17 章）。主要介绍 Ajax 技术与 PHP 结合的简单实际应用。首先介绍如何处理服务器响应，然后讲解 Ajax 技术的设计模式，接下来，以实例的形式详细介绍 Ajax 技术的使用技巧，例如在 PHP 页面上显示进度条、级联菜单形式显示数据库数据，以及 PHP+Ajax 数据分页显示等。

第 5 篇：综合实例篇（第 18 章）。本实例实现一个教学视频网站后台管理系统，让读者全面掌握如何在实际项目中使用 Ajax 技术，并提高读者对大型应用系统的整体把握，同时提高读者对 PHP+Ajax 技术的熟练程度。

基本上囊括了 PHP 开发的基础知识、Ajax 开发的基础知识、PHP+Ajax 组合开发 Web 2.0 程序和 PHP+Ajax 实例程序。

2．本书特色

书中采用大量的实例进行讲解，力求通过实例使读者更形象地理解 PHP 的编程思想，快速掌握基于 Ajax 技术的开发。本书难度适中，内容由浅入深，实用性强，覆盖面广，条理清晰。

- **知识点全** 本书紧紧围绕利用 PHP 与 Ajax 进行 Web 程序开发展开讲解，具有很强的逻辑性和系统性。
- **实例丰富** 书中各实例均经过作者精心设计和挑选，都是根据作者在实际开发中的经验总结而来，几乎涵盖了在实际开发中所遇到的各种问题。
- **应用广泛** 对于精选案例，给了详细步骤，结构清晰简明，分析深入浅出，而且有些程序能够直接在项目中使用，避免读者进行二次开发。
- **基于理论，注重实践** 在讲述过程中，不仅介绍理论知识，而且在合适位置安排综合应用实例，或者小型应用程序，将理论应用到实践当中，来加强读者的实际应用能力，巩固 PHP 开发基础知识。
- **随书光盘** 本书为实例配备了视频教学文件，读者可以通过视频文件更加直观地学习 PHP 和 Ajax 的使用知识。
- **网站技术支持** 读者在学习或者工作的过程中，如果遇到实际问题，可以直接登录 www.itzcn.com，作者会在第一时间给予帮助。

3．读者对象

本书具有知识全面、实例精彩、指导性强的特点，力求以全面的知识性及丰富的实例来指导读者透彻地学习 PHP 和 Ajax 各方面的知识。本书可以作为 PHP Web 开发人员的重要学习资料，也可以作为 PHP 和 Ajax 开发职业培训教程。

本书适合以下人员阅读学习。

- PHP 初学者和自学者。
- 网站后台维护人员。
- PHP 应用开发人员。
- Ajax 前台应用开发人员。
- 各大中专院校的在校学生和相关授课老师。
- 其他 Web 编程爱好者。

除了封面署名人员之外，参与本书编写的还有左旭日、于永军、张秋香、李乃文、张仕禹、夏小军、赵振江、李振山、李文才、吴越胜、李海庆、何永国、李海峰、陶丽、吴俊海、安征、张巍屹、崔群法、王咏梅、康显丽、辛爱军、牛小平、贾栓稳、王立新、苏静、赵元庆、郭磊、徐铭、李大庆、王蕾、张勇、郝安林、郭新志、牛丽平、唐守国等。在编写过程中难免会有疏漏，欢迎读者与编者联系，帮助编者改正提高。

CONTENTS

目录

第 1 篇　PHP 基础篇

第 1 篇　PHP 基础篇

第1章 PHP 入门



内容摘要 Abstract

PHP 是一种被广泛应用的、开放源代码的多用途脚本语言，可以嵌入到 HTML 中，非常适合 Web 开发。PHP 的最大优势是对于初学者来说极其简单，可以很快入门，只需几个小时就可以自己写一些简单的脚本程序，同时也给专业的程序员提供了各种高级的特性。本章将简单介绍 PHP 的产生和发展，并重点介绍构建 PHP 的运行平台，即 Apache 和 PHP 的安装及配置。

学习目标 Objective

- 了解 PHP 的产生和发展
- 了解 PHP 特性
- 掌握如何安装 Apache 和 PHP
- 配置 PHP 运行环境
- 了解 PHP 配置指令
- 开发简单的 PHP 应用程序

1.1 PHP 概述

随着 PHP 的不断改进，越来越多的人意识到它的实用性，PHP 从而逐渐发展起来，并成为开发动态网页的主流技术之一。本节主要介绍 PHP 的产生和发展，还有 PHP 的主要特点。

1.1.1 PHP 产生和发展

PHP 继承自一个名叫 PHP/FI 的工程。PHP/FI 在 1995 年由 Rasmus Lerdorf 创建，最初只是一套简单的 Perl 脚本，用来跟踪访问主页的信息。他给这一套脚本取名为“Personal Home Page Tools”。随着更多功能需求的增加，Rasmus 写了一个更大的 C 语言的实现，它可以访问数据库，可以让用户开发简单的动态 Web 程序。Rasmus 发布了 PHP/FI 的源代码，以便每个人都可以使用它，同时也可以修正它的 Bug，并且改进其源代码。PHP/FI 是专为个人主页 / 表单提供解释程序的程序，已经包含了今天 PHP 的一些基本功能。它有着 Perl 样式的变量，自动解释表单变量，并可以嵌入 HTML，语法本身与 Perl 很相似。但是它很有限，很简单，

还稍微有些不协调。到 1997 年，PHP/FI 2.0，也就是它的 C 语言实现的第二版在全世界已经有几千个用户（估计）和大约 50000 个域名安装，大约是 Internet 所有域名的 1%。但是那时只有几个人在为该工程撰写少量代码，仍然只是一个人的工程。PHP/FI 2.0 在经历了数个 beta 版本的发布后，于 1997 年 11 月发布了官方正式版本。不久，PHP 3.0 的第一个 alpha 版本发布，PHP 从此走向了成功。

PHP 3.0 是类似于当今 PHP 语法结构的第一个版本。Andi Gutmans 和 Zeev Suraski 在为一所大学的项目开发电子商务程序时发现 PHP/FI 2.0 功能明显不足，于是他们重写了代码，这就是 PHP 3.0。经过 Rasmus，Andi 和 Zeev 一系列的努力，考虑到 PHP/FI 已存在的用户群，他们决定联合发布 PHP 3.0 作为 PHP/FI 2.0 的官方后继版本，而 PHP/FI 2.0 的进一步开发几乎终止了。

PHP 3.0 最强大的功能是可扩展性。除了给最终用户提供数据库、协议和 API 的基础结构，它的可扩展性还吸引了大量的开发人员加入并提交新的模块。后来证实，这是 PHP 3.0 取得巨大成功的关键。PHP 3.0 中的其他关键功能包括面向对象的支持、更强大和协调的语法结构。

这个全新的语言伴随着一个新的名称发布。它从 PHP/FI 2.0 的名称中移去了暗含“本语言只限于个人使用”的部分，被命名为简单的缩写“PHP”。这是一种递归的缩写，全称是 Hypertext Preprocessor。

1998 年末，PHP 的安装人数接近 10000，有大约 100000 个网站报告他们使用了 PHP。在 PHP 3.0 的顶峰阶段，Internet 中 10%的 Web 服务器上都安装了它。约 9 个月的公开测试后，官方于 1998 年 6 月正式发布 PHP 3.0。

PHP 代表超文本预处理器（PHP，Hypertext Preprocessor）。PHP 是完全免费的，可以从 PHP 官方站点（http://www.php.net）下载。PHP 遵守 GNU 公共许可（GPL），在这一许可下诞生了许多流行的软件诸如 Linux 和 Emacs。用户可以不受限制地获得源码，甚至可以从中加进自己需要的特色。

1.1.2 PHP 介绍

PHP 是一种广泛应用的开源多用途脚本语言，它可以嵌入到 HTML 中，尤其适合 Web 开发。PHP 是一种基于服务器端的脚本语言，可以用来完成任何其他的 CGI 程序能够完成的工作，例如收集表单数据，生成动态网页，或者发送 / 接收 Cookies。使用 PHP 还有很多好处，如实用性、对多种数据库的支持、对网络协议的支持、面向对象编程、跨平台性和可扩展性等。下面简要介绍这些特性。

1. 支持多种数据库

PHP 最强大的特性是支持多种数据库，例如 Oracle、SQL Server 和 MySQL 等。其中 PHP 与 MySQL 是最佳组合。由于 PHP 支持 ODBC（开放数据库连接标准），因此 PHP 可以连接任何其他支持该标准的数据库。

2. 支持多种网络协议

PHP 支持 LDAP、IMAP、SNMP、NNTP、POP3、HTTP 和 COM 等大量协议。PHP 支持

对 Java 对象的即时连接，并且可以将其自由地用作 PHP 对象，甚至还可以用 CORBA 扩展库来访问远程对象。

3．面向对象编程

使用 PHP 进行 Web 开发时，由于 PHP 提供了类和对象，因此可以选择面向对象方式编程，当然也可以选择面向过程方式编程，或者选择两者混合的方式。

4．文本处理功能

PHP 支持 POSIX 扩展、Perl 正则表达式和 XML 文档解析。PHP 之所以能够解析 XML 文档，是因为 PHP 还支持 SAX 和 DOM 标准，可以使用 XSLT 扩展库来转换文档。PHP 还可以输出文本，例如 XHTML 和其他形式的 XML 文件。

5．可扩展性

在 PHP 应用程序中，程序员可以为 PHP 扩展附加功能。例如程序员可以为应用程序添加一个扩展类。

6．跨平台性

PHP 跨平台性非常好，例如在 Linux 平台、GUN/Linux 和 Windows 平台上都可以运行。

上面介绍了 PHP 脚本语言的一些特性，下面详细介绍 PHP 脚本语言的一些具体运用。一个有用的 PHP 脚本可能只包含一行代码，与 C 语言不同，不需要导入函数库。例如下面的代码就是一个完整的 PHP 脚本，其目的是以类似于 September 23，2005 的格式输出当前的日期：

```
<?php echo date("F j,y");?>
```

PHP 语言很强调紧凑性，这反映在 PHP 能嵌套函数。例如，通过在一行代码中按特定的顺序调用函数，可以对一个值进行一系列修改。下面的例子将生成一个由字母或者数字字符组成的伪随机字符串，如 a3jh8：

```
$randomString=substr(mds(microtime()),0,5);
```

PHP 是一种弱类型的语言，即类型松散的语言，这意味着不需要明确地创建变量，指派类型或撤销变量，当然也没有绝对禁止做这些操作。PHP 在内部处理这些情况，脚本中使用变量时 PHP 会动态创建变量，并使用最优推测规则自动指派变量的类型。例如，PHP 认为代码 1.1 的语句完全合法。

代码 1.1　代码片段

```
$string_value="你好";
echo($string_value);
$radius=2.0;
$pi=3.14159;
$area=$pi*$radius*$radius;
```

PHP 还会在脚本结束时自动撤销变量，将资源返回给系统。从这些方面来看，PHP 在内

部处理了编程的许多管理方面的问题，这就允许开发人员集中精力去完成最终的目标。

1.2 搭建 PHP 环境

PHP 在 Apache 服务器中能够快速运行，所以要首先创建一个 Apache 服务器，然后再为 PHP 配置环境。本节主要讲解如何安装 Apache 服务器、安装 PHP 和配置计算机 PHP 环境。

1.2.1 安装 Apache 和 PHP

只有在安装了 Apache 服务器和 PHP 之后才能调试运行 PHP 程序。下面详细讲解如何下载和安装 Apache 和 PHP。

1. 安装 Apache

Apache 是当今在 IT 界非常受欢迎的 Web 服务器，并且 Apache 的版本更新速度非常快，还可以在多种系统上安装。需要安装此程序，只需从官方网站下载安装程序。打开浏览器在地址栏中输入官方网址http://www.apache.org转到该网站，然后在该网站中下载最新版本的 Apache。在该网站的首页单击 HTTP Server 超链接，转到 Apache 服务器介绍页面，如图 1-1 所示。

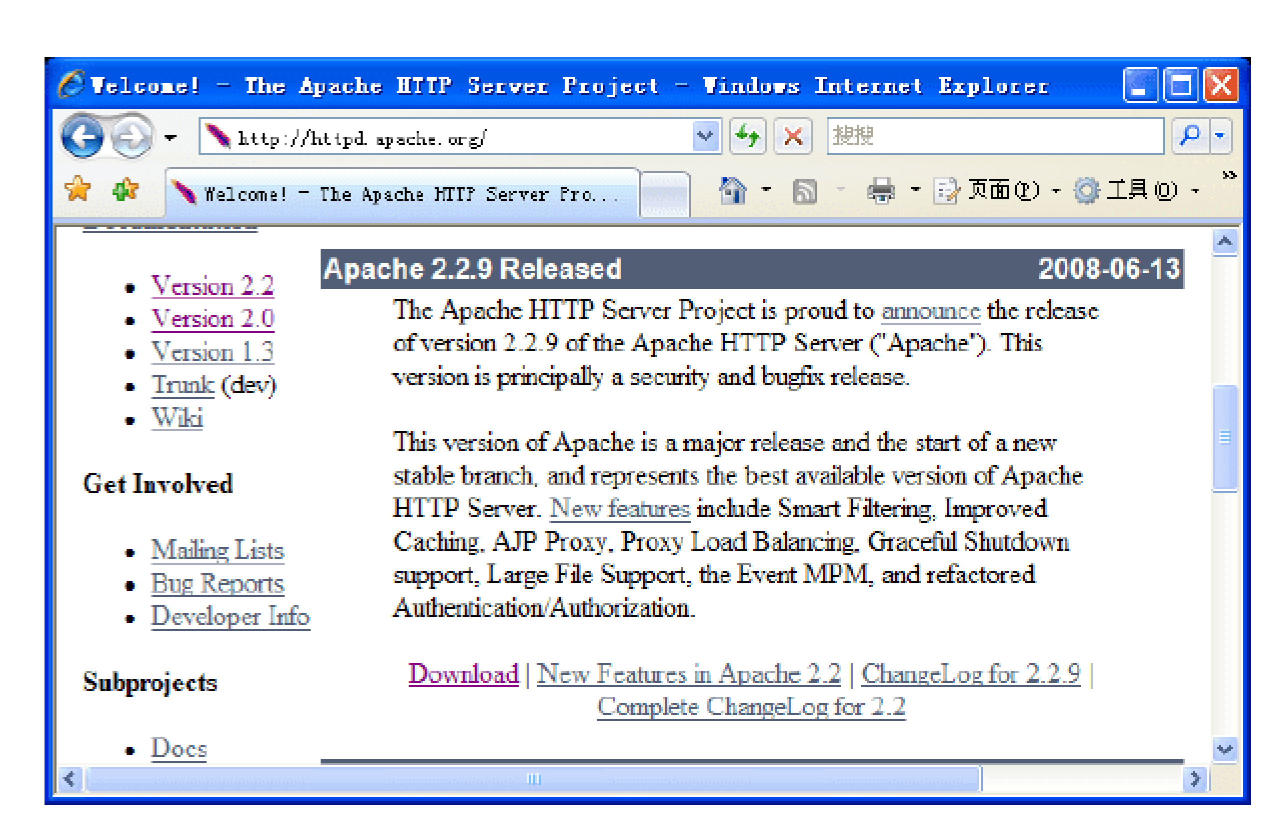


图 1-1 Apache 2.2.9 显示页面

然后单击 Download 超链接，转到 Apache 安装程序下载页面，在该页面上提供了 5 种下载程序。由于是在 Windows 系统下配置和调试 PHP，因此需要下载 Win32 系列，这里包含两种形式的安装程序：一种是以源代码的形式存在，另一种是以二进制安装程序的形式存在。在此下载二进制可执行文件，该下载页面效果如图 1-2 所示。

此时，就下载好了 Apache 安装程序，双击 apache_2.2.9-win32-x86-no_ssl-r2.msi 安装程序，启动 Apache 服务器安装向导。图 1-3 所示为 Apache 服务器欢迎安装对话框，然后单击 Next 按钮显示如图 1-4 所示的对话框。

选择 I accept the terms in the license agreement 单选按钮，此时就激活了 Next 按钮，会显

示图 1-5 所示的对话框，该对话框主要介绍了 Apache 服务器。单击图 1-5 中的 Next 按钮，弹出 Server Information 向导对话框，如图 1-6 所示。

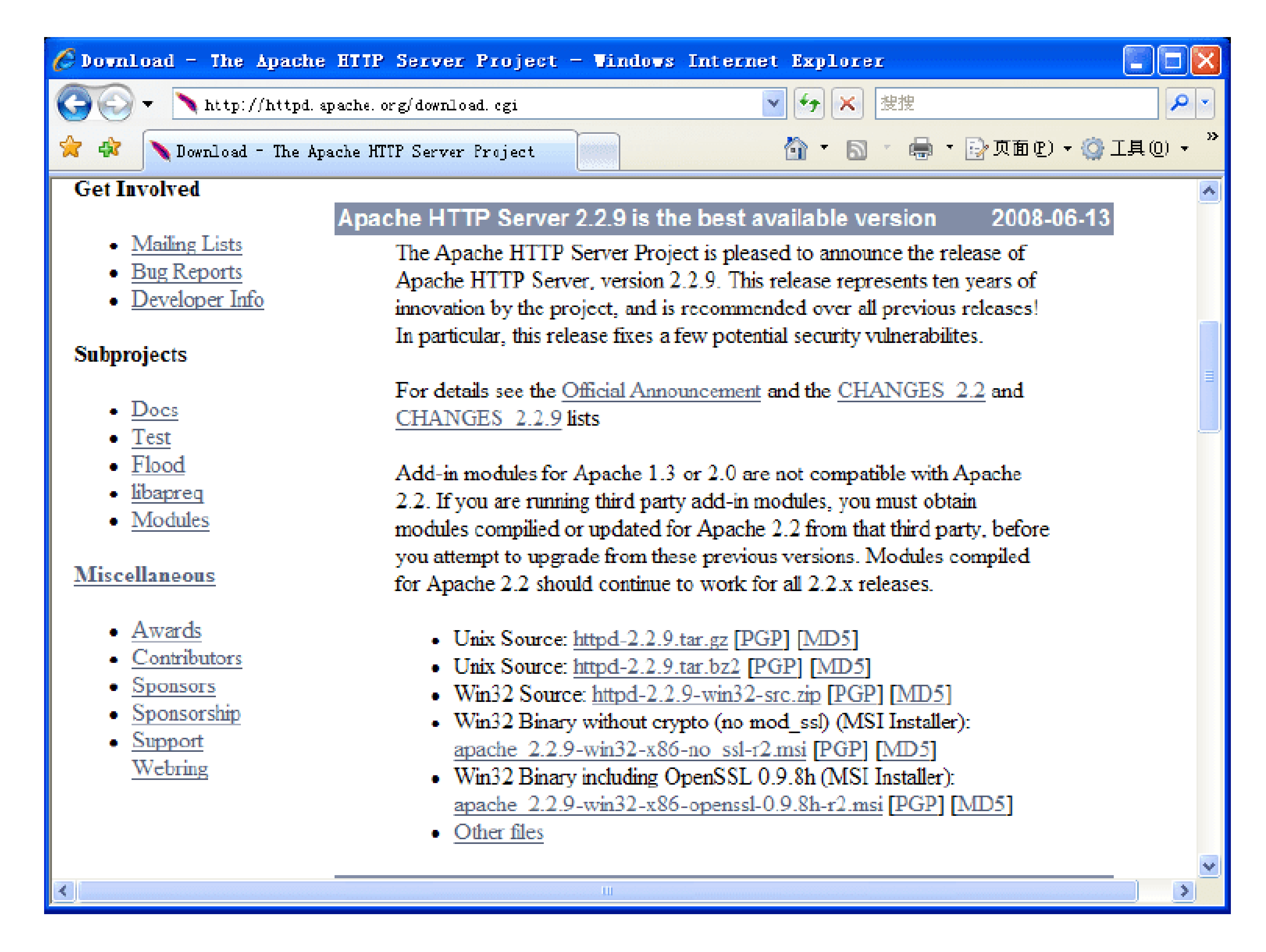


图 1-2　Apache 安装程序下载页面

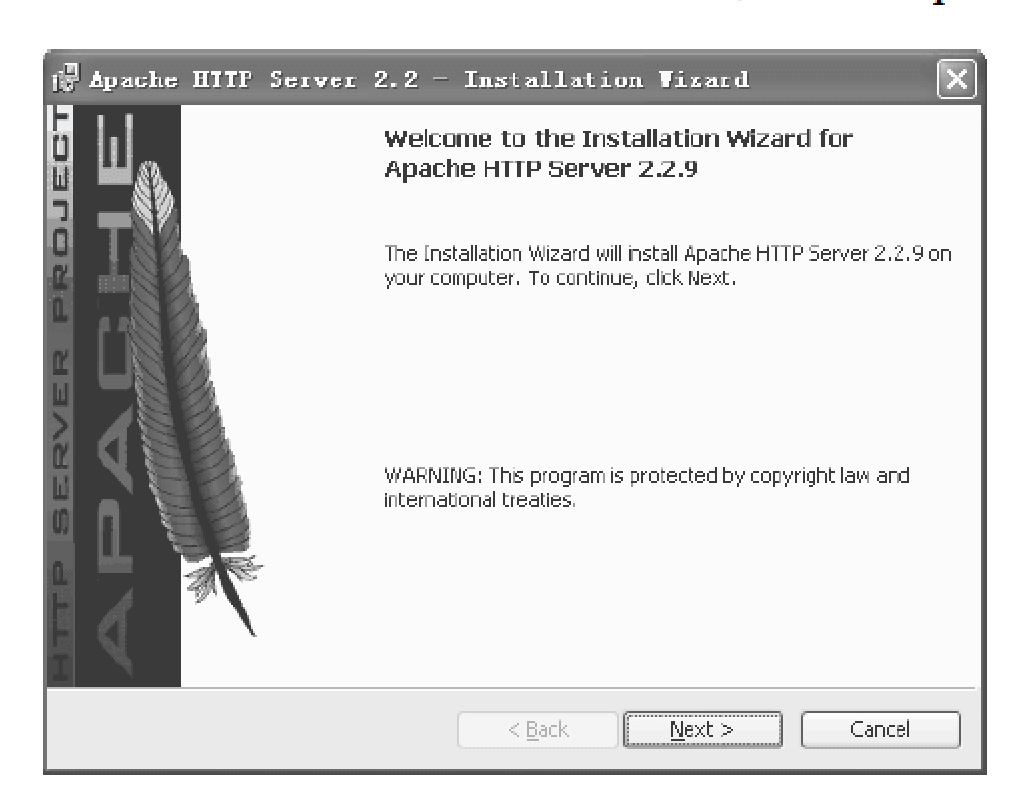


图 1-3　Apache 安装程序起始对话框

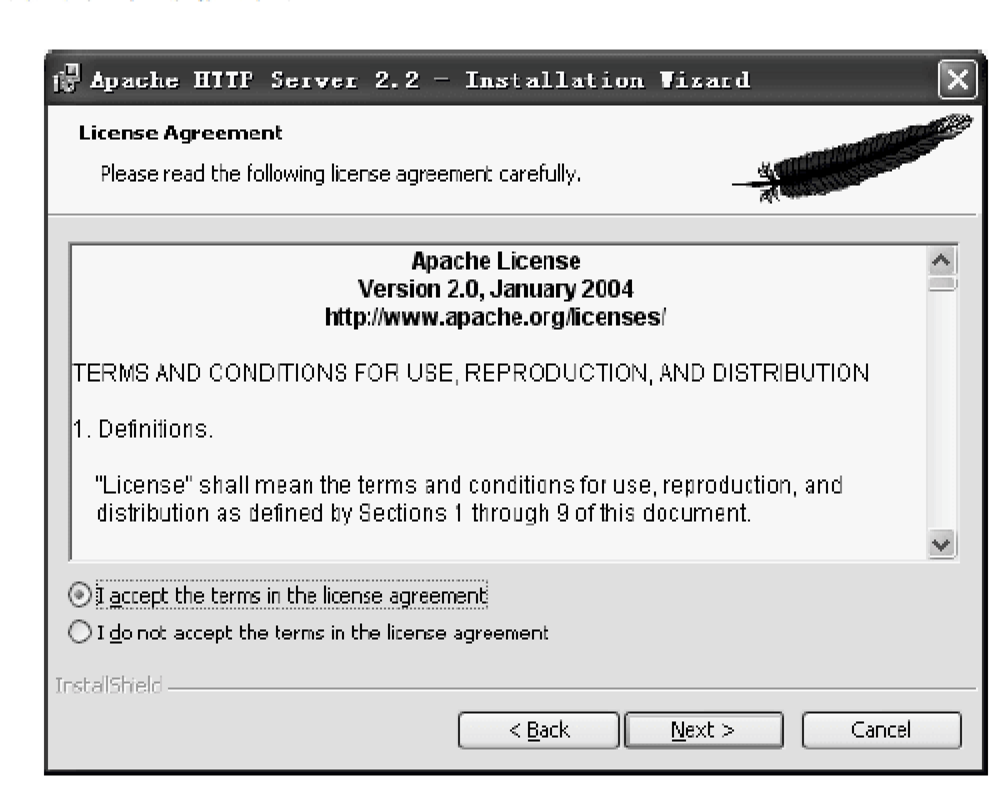


图 1-4　安装程序接受协议对话框

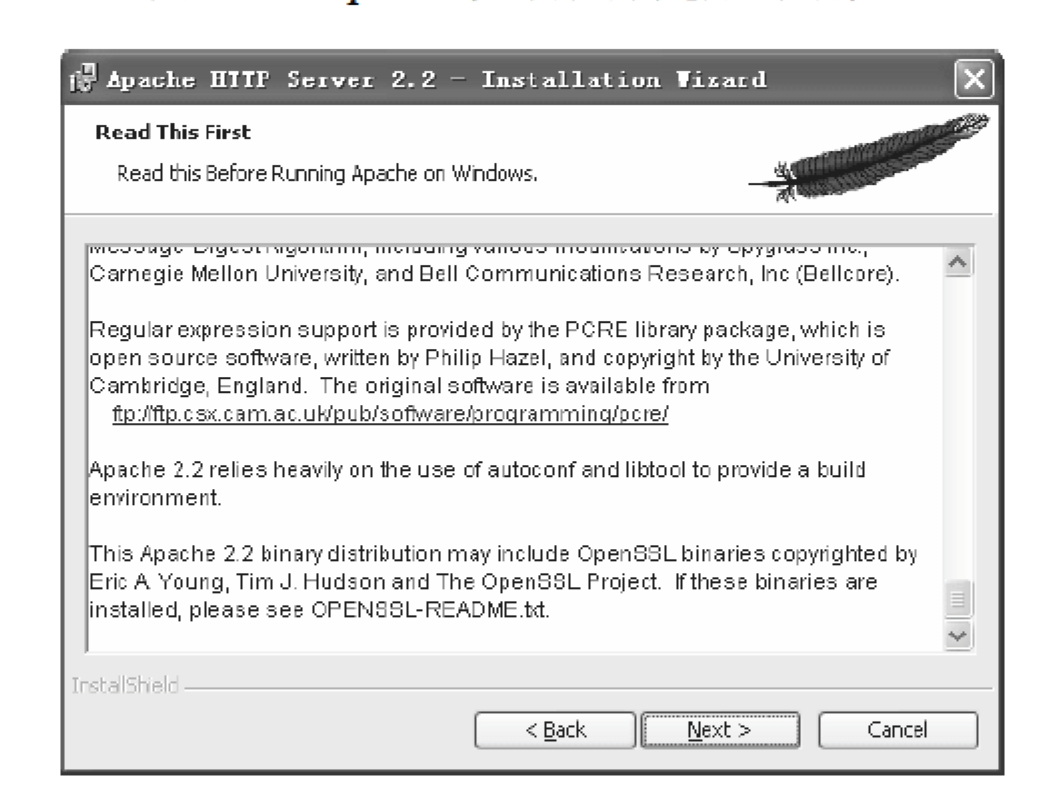


图 1-5　Apache 服务器介绍对话框

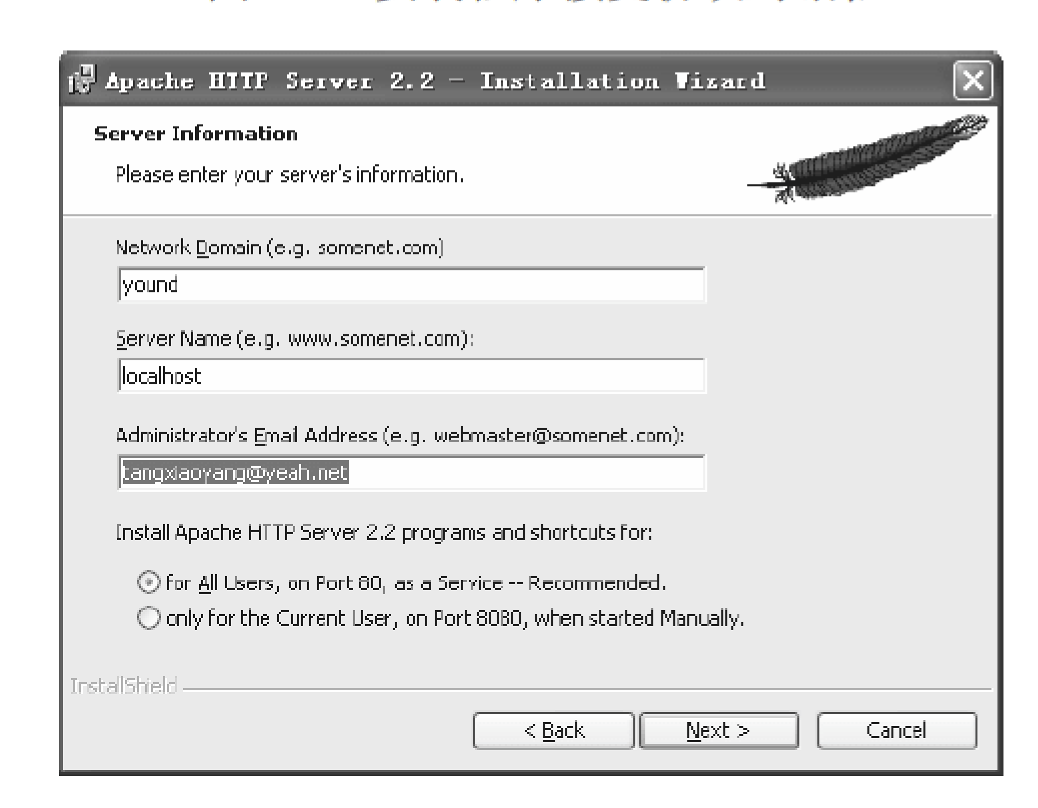


图 1-6　输入 Apache 服务器选项

在图 1-6 中，第一个文本框要求输入计算机的网络域，如果没有填写任何值，表示本机 IP 地址；第二个文本框表示服务器的名称，这里可以设定为 localhost；第三个文本框是系统管理员的电子邮件地址，在这里可以输入自己的邮件地址，以后再进行修改。在下面安装程序会提示希望为所有用户提供 Apache 服务，还是仅为当前用户提供服务，这里选中 for All Users，on Port 80，as a Server--Recommended.单选按钮。单击 Next 按钮，会出现如图 1-7 所示的对话框，在对话框中可以选择安装的类型，这里有典型安装和定制安装，本文选择典型安装，在该对话框中单击 Next 按钮弹出图 1-8 所示对话框。

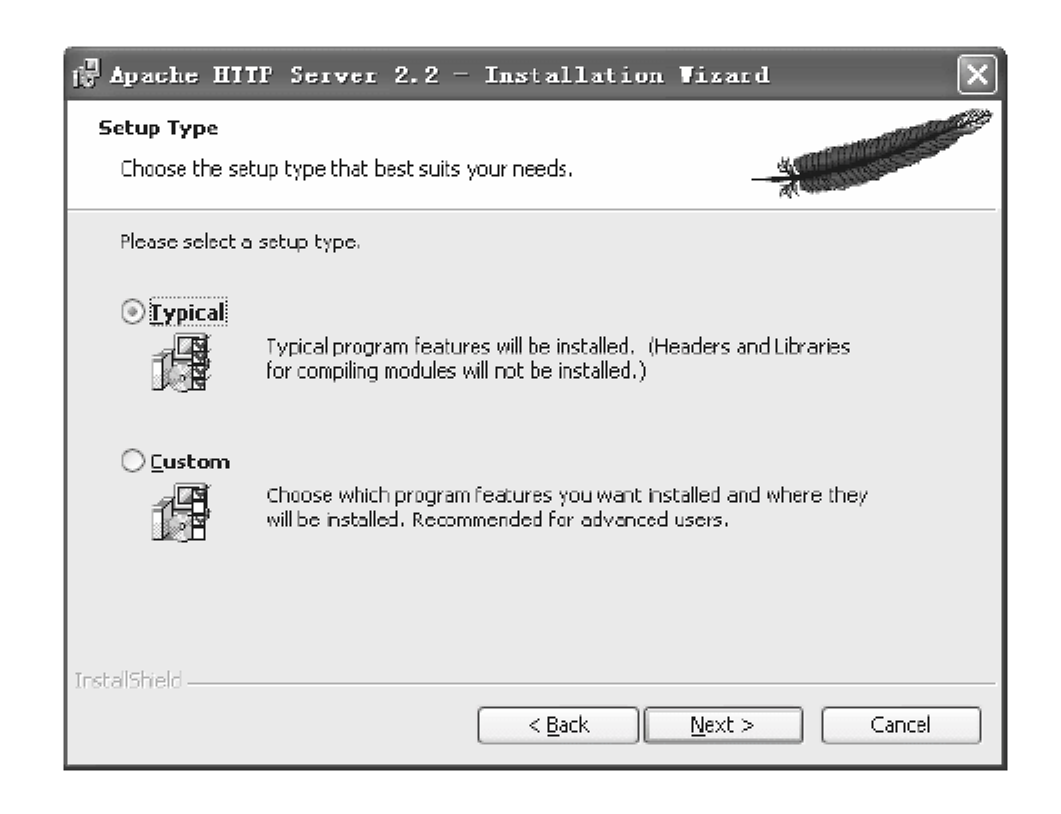



图 1-7　安装类型对话框

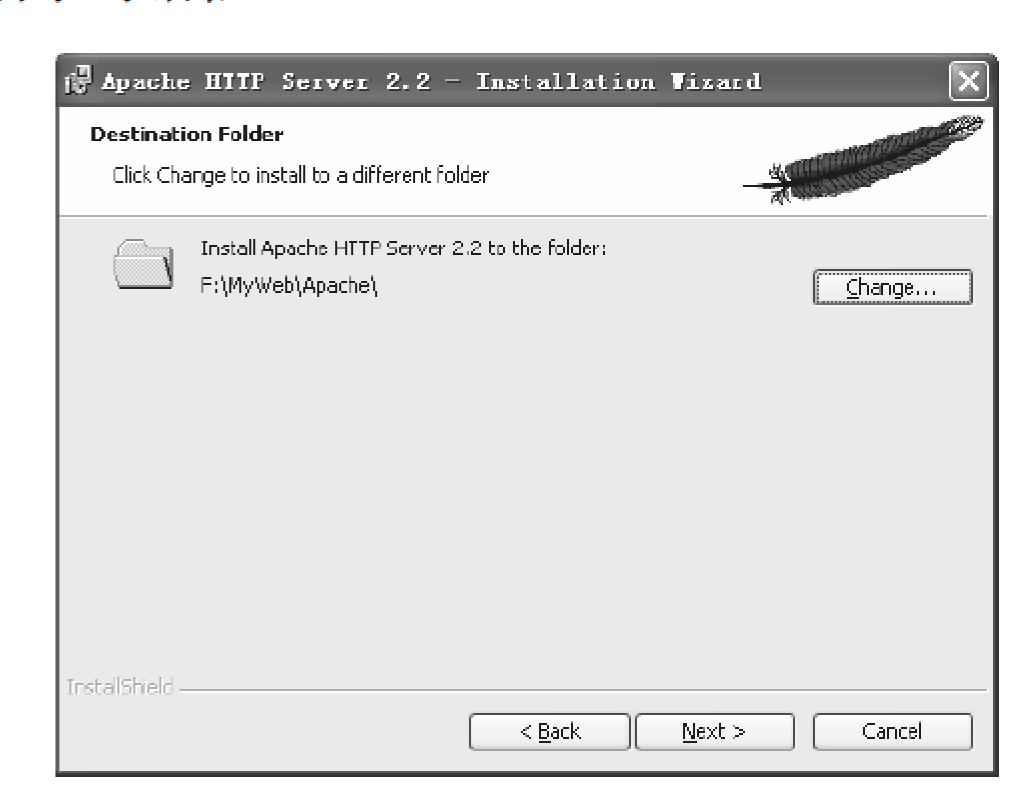


图 1-8　安装路径对话框

在图 1-8 中单击 Change 按钮，在该对话框中可以自由设定 Apache 的安装目录，这里选择的是 F:\MyWeb\Apache 目录。单击 Next 按钮会显示图 1-9 所示的对话框，单击 Install 按钮，开始安装 Apache 服务器，安装效果如图 1-10 所示的窗口。

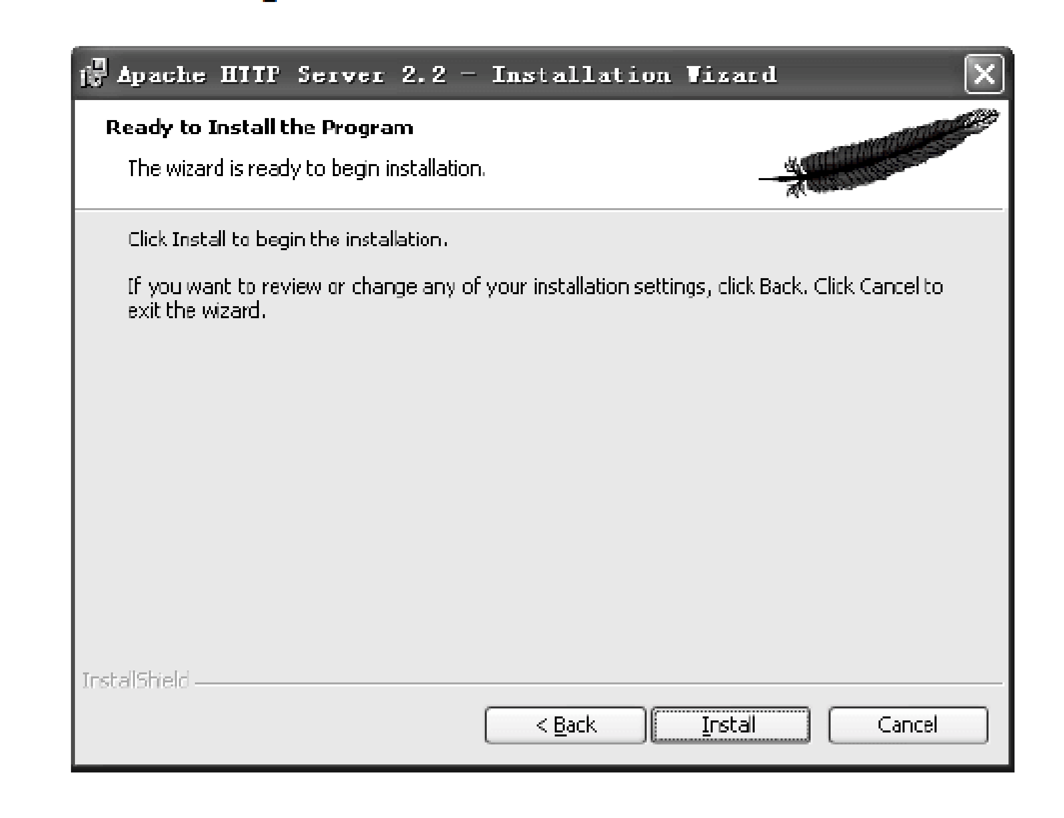


图 1-9　准备安装

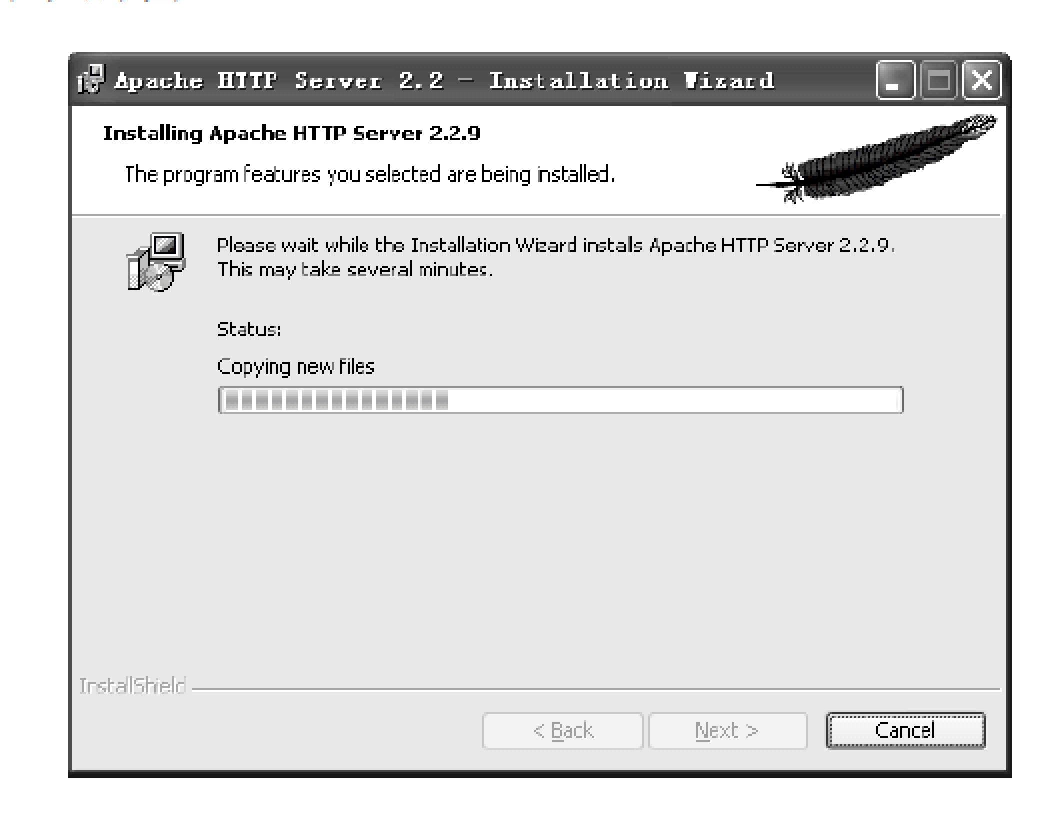


图 1-10　开始安装

安装完成之后，会出现一个提示对话框，如图 1-11 所示，在对话框中会提示成功安装了 Apache 服务器。

这时会在右下角的状态栏出现图标，表示 Apache 已经成功安装了。单击该图标，会出现图标，单击该图标会出现图标，在此处单击 Start 菜单项就可以启动 Apache 了。成功启动后，会出现图标。此时 Apache 服务器就安装完成了。下面检验 Apache 是否安装成功，打开 IE 浏览器，在地址栏中输入 http://localhost/，单击【转到】按钮，会显示如

图 1-12 所示的页面。

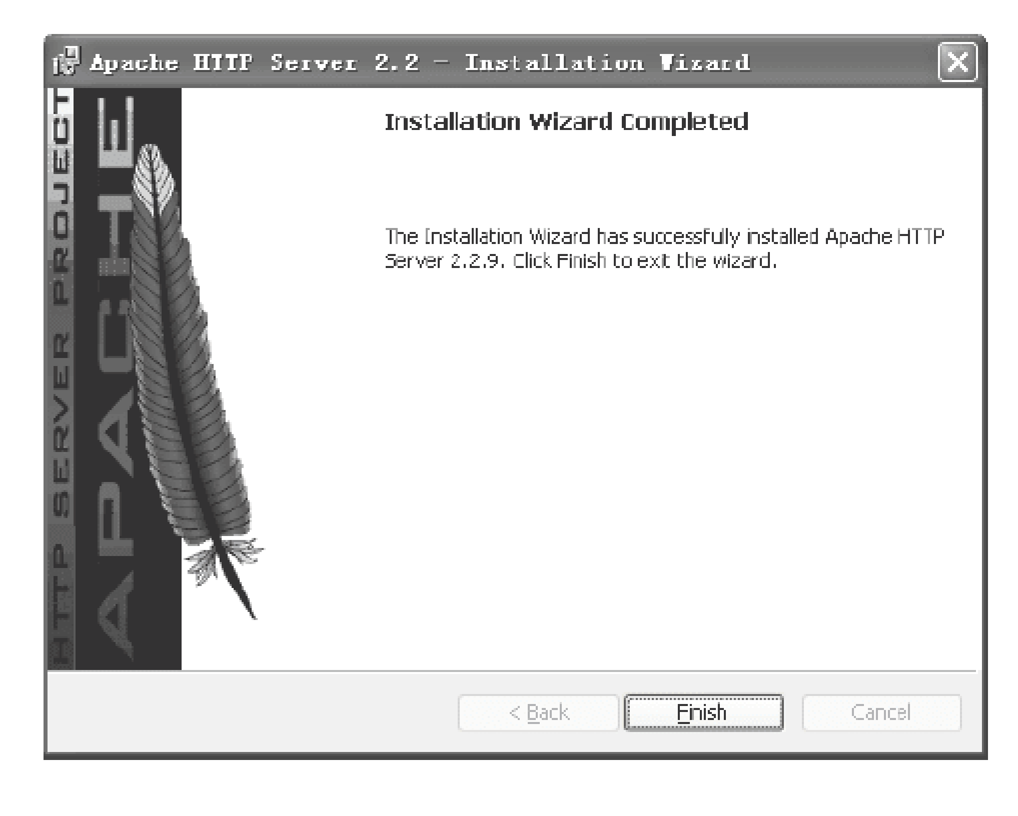


图 1-11　安装成功

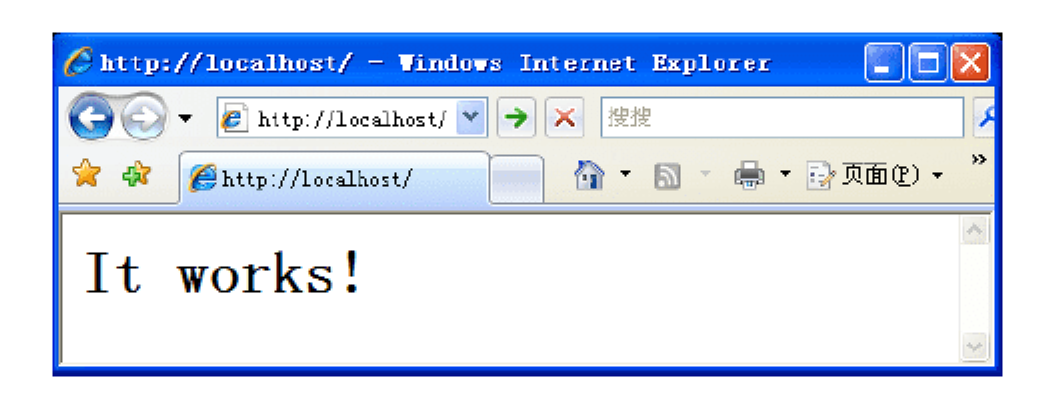


图 1-12　Apache 正常启动

2. 安装 PHP

PHP 开发工具包是开发 PHP 程序的核心，该工具包需要从 PHP 官方网站上下载。只有安装了该开发工具包，才能解释执行 PHP 页面的脚本程序，例如执行 PHP 页面函数。打开 IE 浏览器，在地址栏中输入http://www.php.net打开网站主页，页面效果如图 1-13 所示。

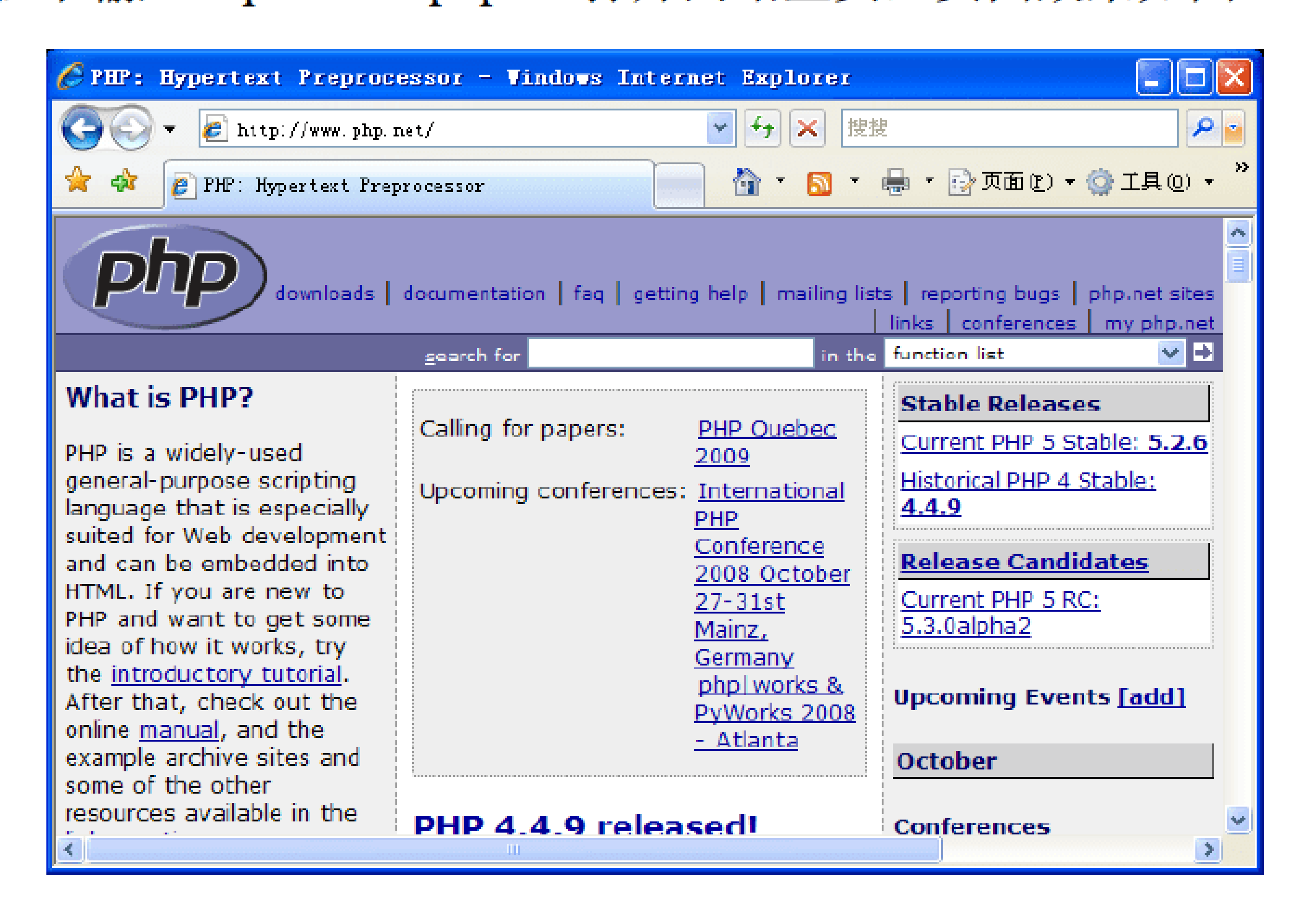


图 1-13　PHP 官方网站主页

然后单击 downloads 超链接，转到下载 PHP 工具包的页面。当然也可以下载二进制可执行程序，该页面效果如图 1-14 所示，单击下载链接后，转到下载工具包的页面，如图 1-15 所示。

可以在该页面上下载工具包，也可以下载安装程序。本文下载了工具包和安装程序，安装工具包时只需要解压缩工具包即可。本文主要讲解如何使用 PHP 安装程序安装向导。

双击下载好的 php-5.2.6-win32-installer.msi，弹出该 PHP 安装程序的安装向导窗口，如图 1-16 所示。

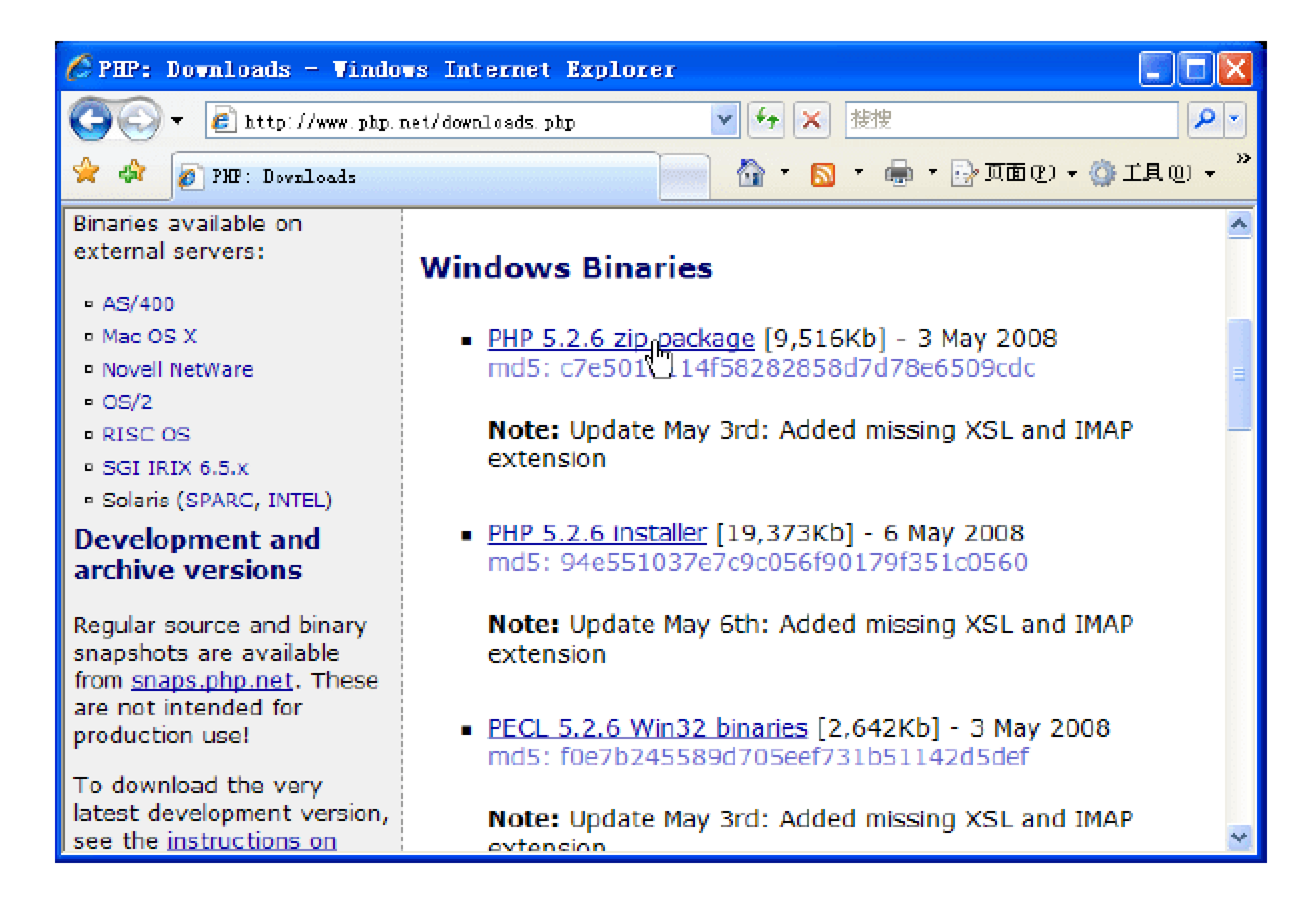


图 1-14　下载工具包信息页面

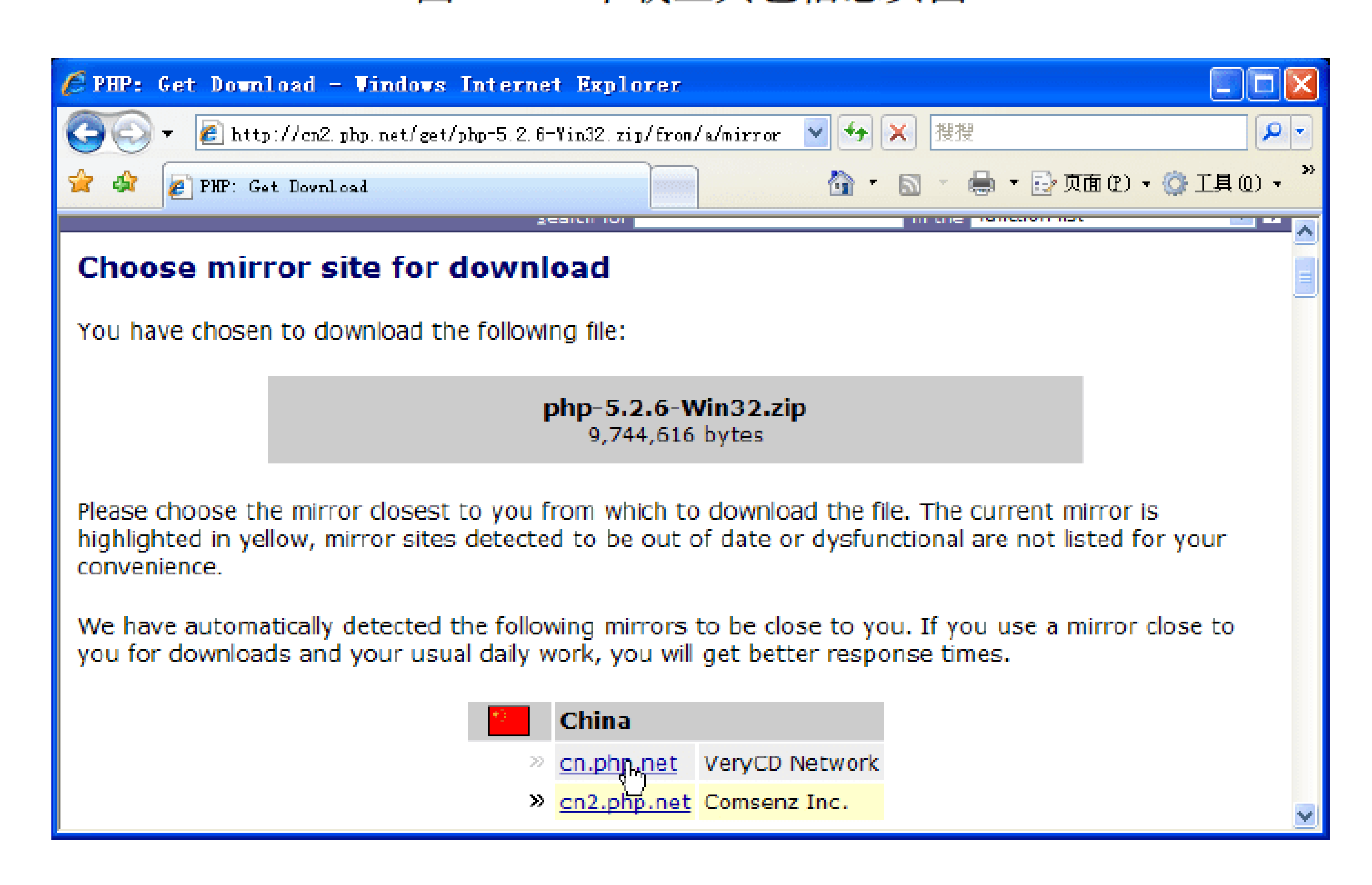


图 1-15　下载页面

单击图 1-16 中的 Next 按钮，弹出用户协议窗口，如图 1-17 所示。在该用户协议向导中选中复选框，激活 Next 按钮，然后单击 Next 按钮，弹出为 PHP 工具包选择路径的窗口，如图 1-18 所示。



图 1-16　PHP 安装程序安装向导窗口

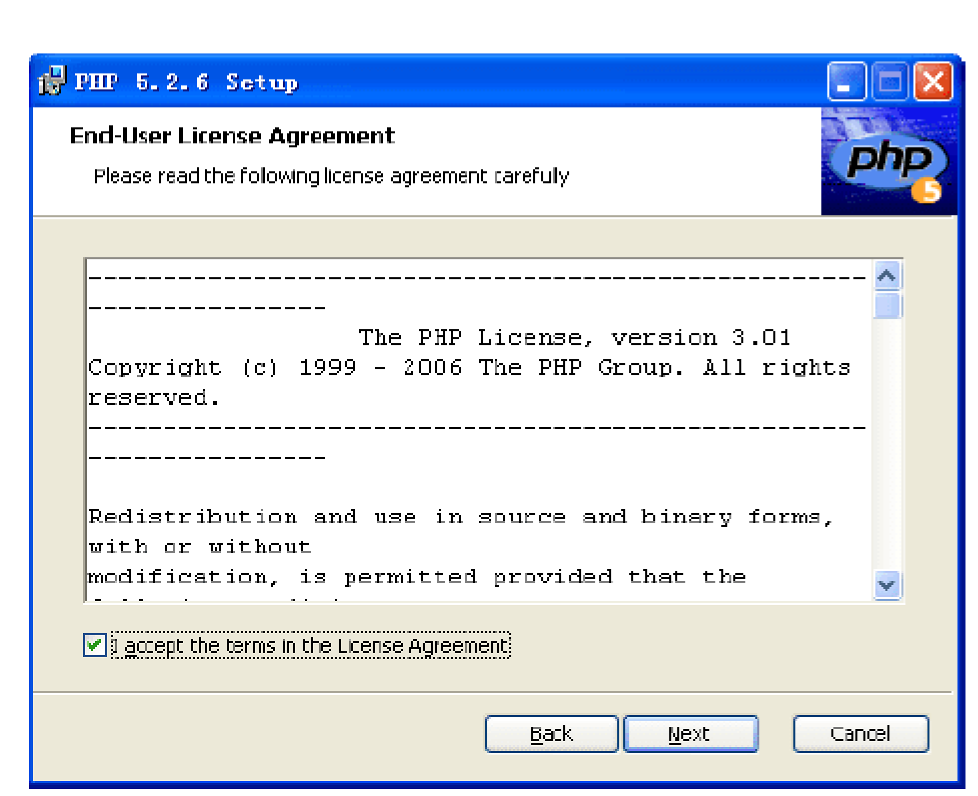


图 1-17　安装向导用户协议窗口

在图 1-18 中，为 PHP 选择好路径后，接下来单击 Next 按钮，弹出选择 Apache Web 服务器的窗口，选择安装 Apache 版本号后的窗口，如图 1-19 所示。

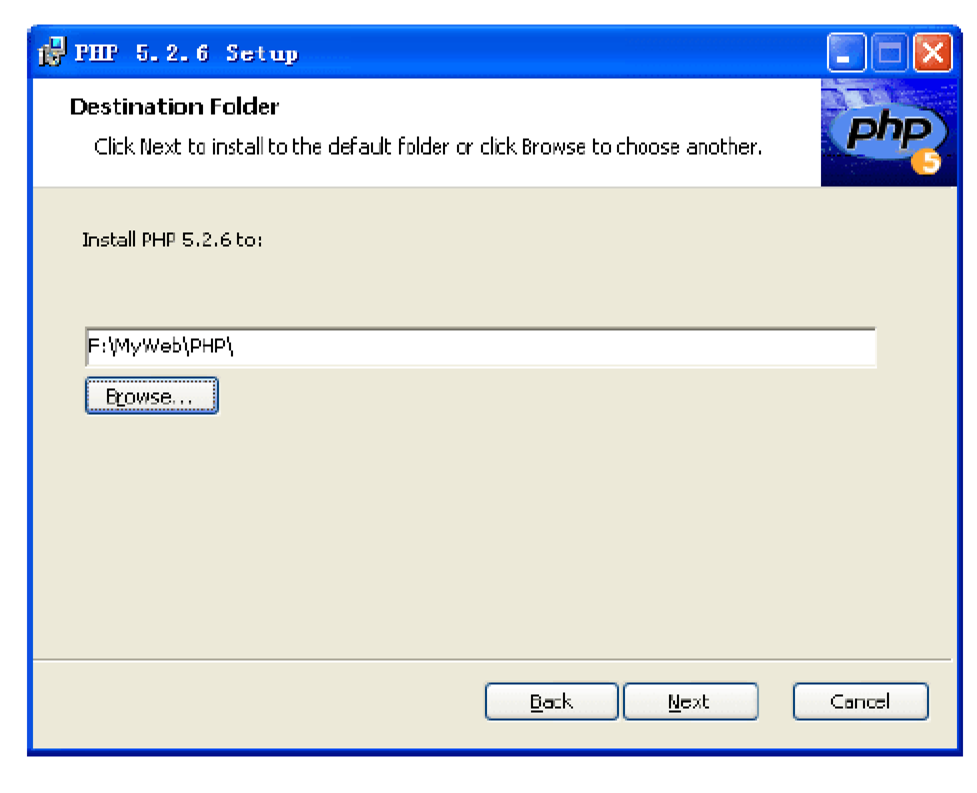



图 1-18　选择路径窗口

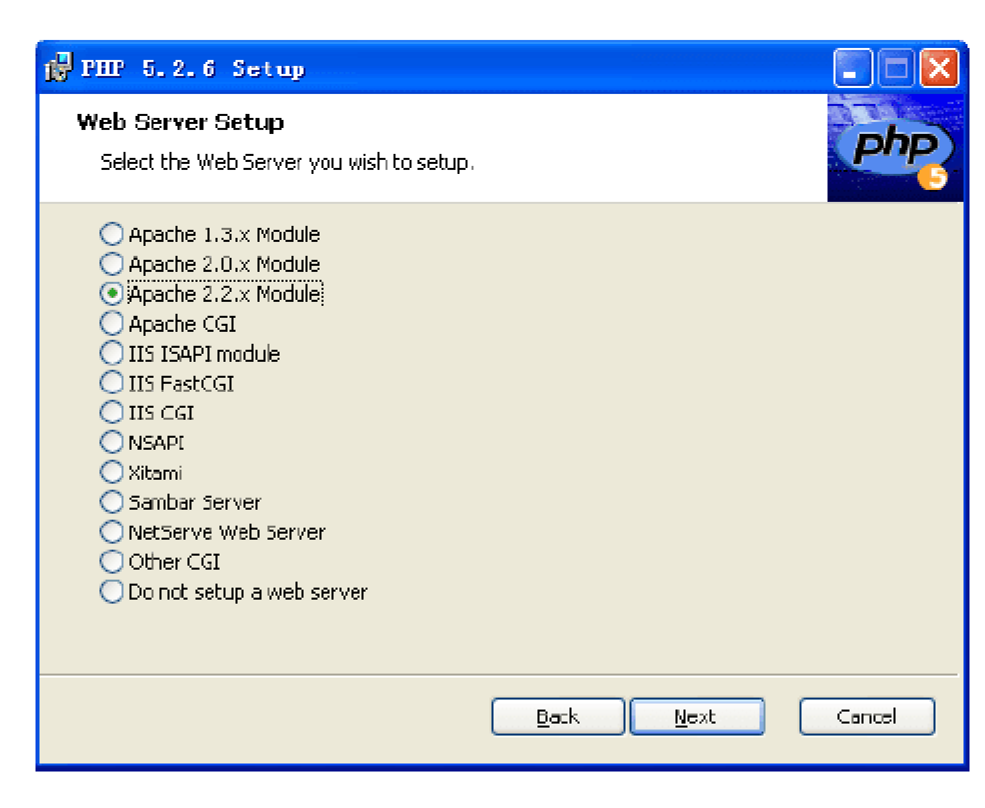


图 1-19　安装向导选择 Web 服务器版本窗口

在图 1-19 中选择好 Apache Web 服务器版本后，单击 Next 按钮，弹出选择 Apache 服务器的安装路径窗口，如图 1-20 所示。

在图 1-20 中，选择好 Apache 服务器的安装路径，然后单击 Next 按钮，弹出选择 PHP 组件安装窗口，如图 1-21 所示。在该窗口中选择安装所有的组件，然后单击 Next 按钮，弹出 PHP 工具包安装窗口，如图 1-22 所示。

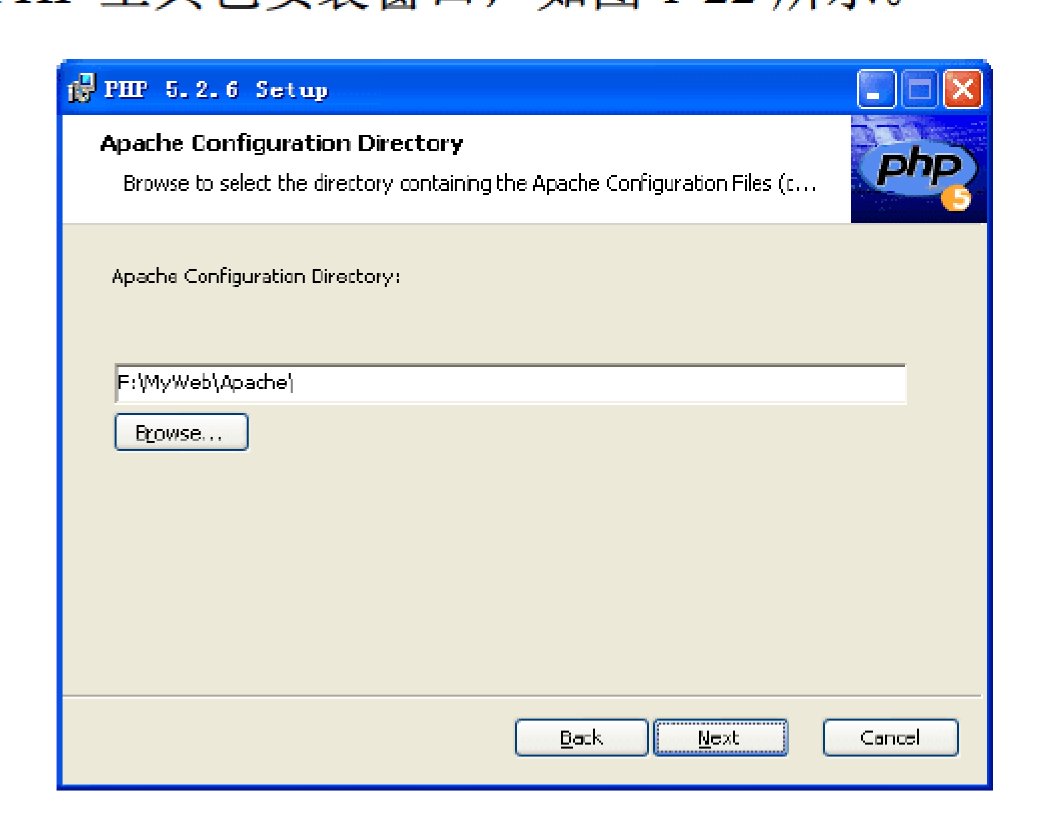


图 1-20　安装向导选择 Web 服务器安装路径窗口

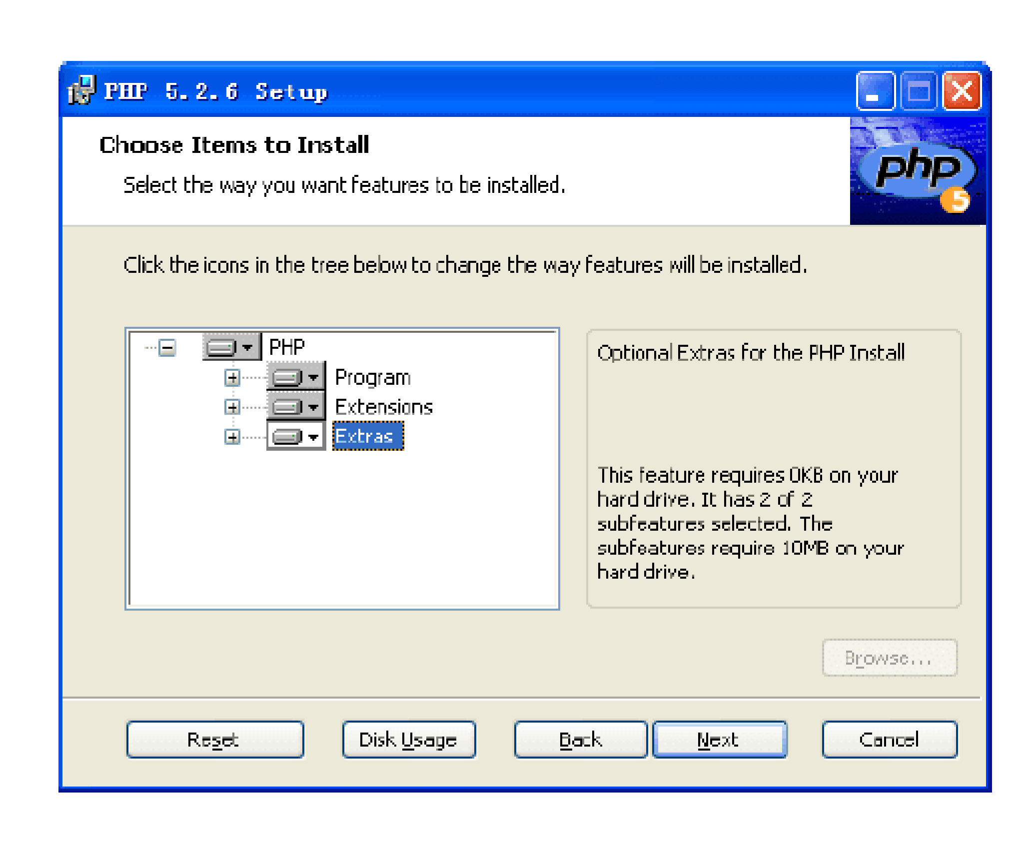
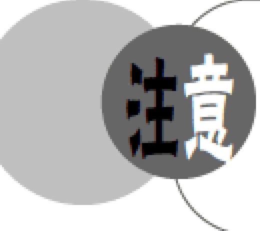


图 1-21　选择组件窗口

安装完成后，会显示如图 1-23 所示的完成窗口，提示成功安装了 PHP 工具包，最后单击 Finish 按钮完成安装。

PHP 工具包安装完成后，就可以在 Apache 中测试.php 文件了。如果没有使用 PHP 工具包安装程序安装，而是直接解压缩工具包，那么需要设置计算机的环境变量，设置环境变量的具体方法如下所示。

（1）解压缩工具包。将工具包解压缩到指定文件夹中，例如“F:\MyWeb\PHP”目录下。



在选择解压缩路径时，要注意不能使用带有空格的路径，例如“F:\Program Files\PHP”。

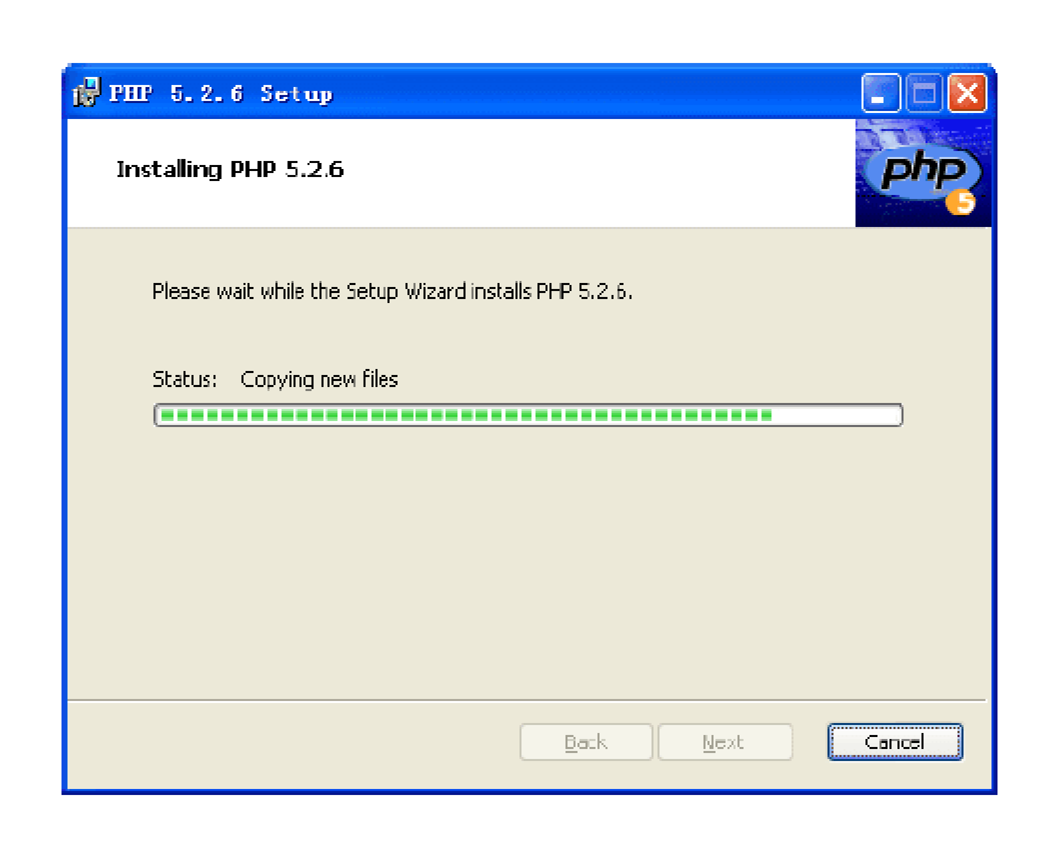


图 1-22　PHP 工具包安装窗口

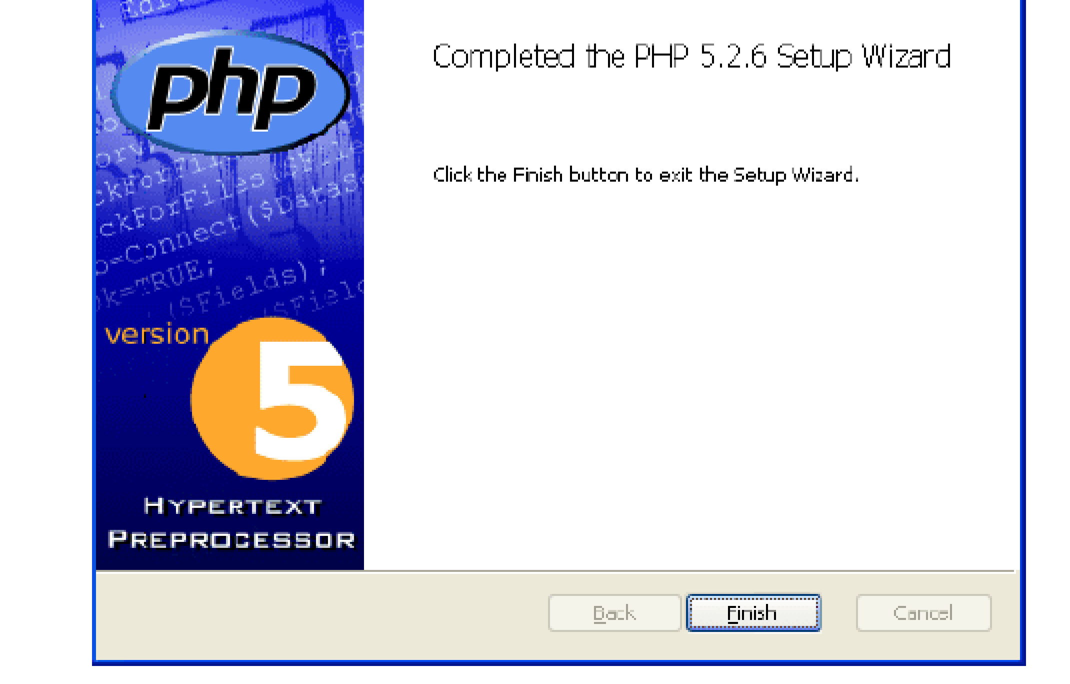


图 1-23　安装向导完成安装窗口

（2）配置 Apache 运行时需要加载的 php5apache2_2.dll 文件。最简单的方法是将 PHP 的安装路径追加到 Windows 系统中 path 路径的下面。右击【我的电脑】，选择【属性】命令，在弹出的对话框的【高级】选项卡中单击【环境变量】按钮，在【环境变量】对话框中找到 Path 路径，单击【编辑】按钮，在【编辑系统变量】对话框中将 F:\MyWeb\PHP 追加到路径中，如图 1-24 所示。



图 1-24　修改 path 路径

（3）打开 F:\MyWeb\Apache\conf 目录，找到 httpd.conf 文件，打开该文件，在文件的最后面增加 3 行内容，如代码 1.2 所示。

代码 1.2　在 httpd.conf 文件中添加内容

```
LoadModule php5_module "F:\MyWeb\php\\php5apache2_2.dll"
AddType application/x-httpd-php .php
PHPIniDir "F:\MyWeb\php"
```

在该段代码中，第一行表示要加载的模块在哪个位置存储，第二行表示将一个 MIME 类型绑定到某个或某些扩展名。.php 只是一种扩展名，这里可以设定为.html、.php2 等。第三行表示 PHP 所在的初始化路径。此时 PHP 环境就配置完成了。

1.2.2　测试 PHP 环境

在本节中将创建一个 PHP 示例，测试 PHP 工具和 Apache 服务器是否安装成功。该示例是执行一个带有 PHP 脚本的程序，如果正确运行则证明 PHP 工具安装成功，否则说明安装失败。在目录“F:/MyWeb/Apache/htdocs”中创建一个记事本文档，并修改其名称为 info.php，在该文件中输入如下所示代码：

```
<?php
    phpinfo();
?>
```

保存好 info.php 文件后，打开 IE 浏览器，在地址栏中输入 http://localhost/info.php，单击【转到】按钮，打开测试页面。如果显示相关信息，则证明 PHP 工具包和环境配置成功，否则失败，图 1-25 为配置成功情况下 info.php 页面的效果。

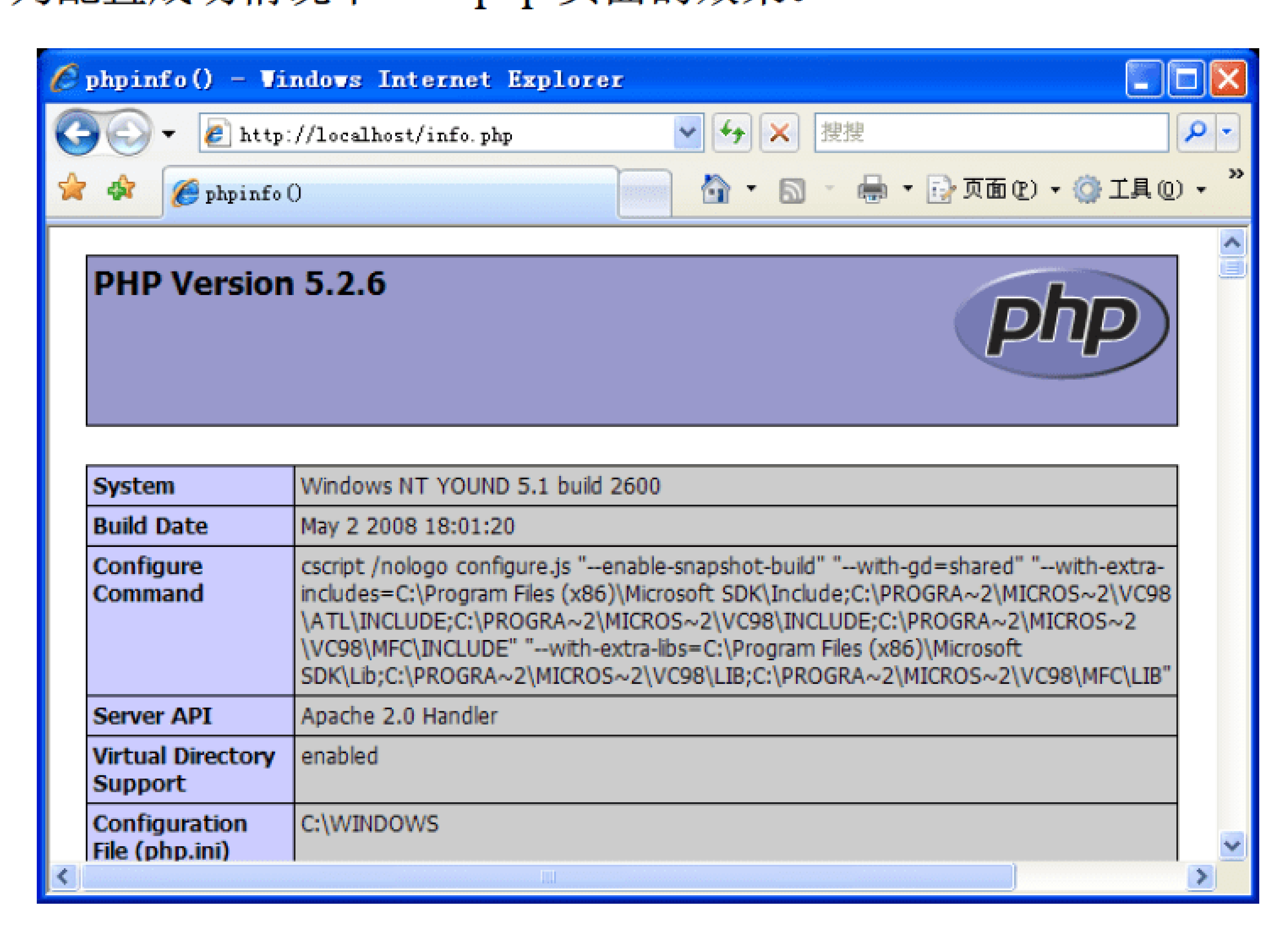


图 1-25　info.php 页面效果图



phpinfo()函数提供与 PHP 安装相关的一组信息。

1.2.3　使用 PHP 帮助文档

PHP 帮助文档介绍了 PHP 的相关知识，当对 PHP 功能有疑问时，可以及时查询 PHP 帮助文档。该文档可以在线查看，也可以下载到计算机中随时查看。本节主要讲解如何从官方网站下载 PHP 帮助文档和查看帮助文档。

打开 IE 浏览器，在地址栏中输入 http://www.php.net，单击【转到】按钮，打开 PHP 官方网站，如图 1-13 所示。在官方网站中单击 documentation 超链接，会转到选择不同国家语言的帮助文档页面，该页面效果如图 1-26 所示。

然后在该页面选择一种语言，例如选择 English，就可以在页面上查看英文的 PHP 帮助文档，如图 1-27 所示。

如果要下载 PHP 帮助文档，只需在图 1-26 中单击 documentation downloads 超链接，转到 PHP 帮助文档下载页面，在下载页面中选择一种语言，然后下载。下载页面如图 1-28 所示。



图 1-26 语言类型选择页面

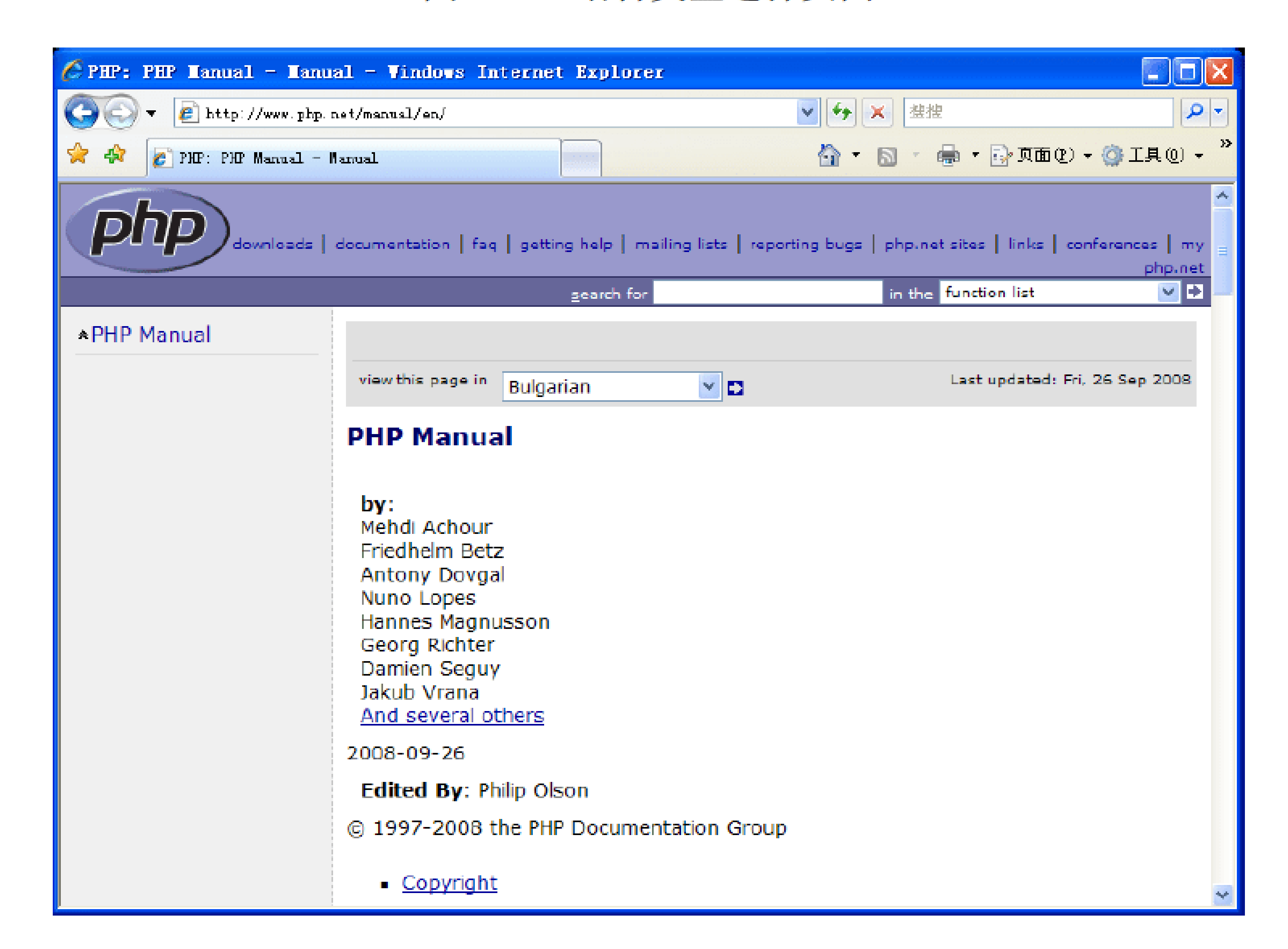
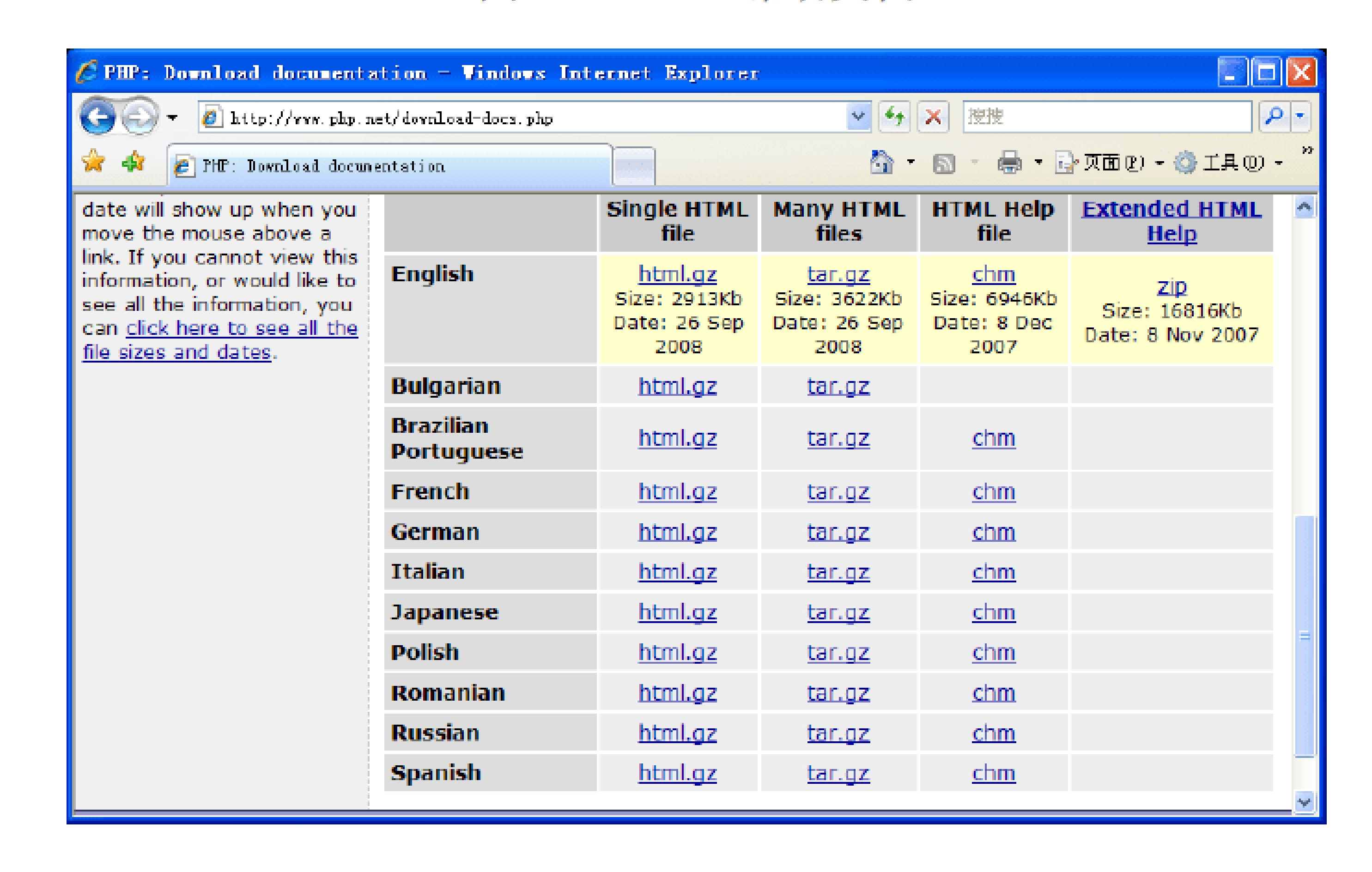


图 1-27 PHP 帮助文档



	Single HTML file	Many HTML files	HTML Help file	Extended HTML Help
English	html.gz Size: 2913Kb Date: 26 Sep 2008	tar.gz Size: 3622Kb Date: 26 Sep 2008	chm Size: 6946Kb Date: 8 Dec 2007	zip Size: 16816Kb Date: 8 Nov 2007
Bulgarian	html.gz	tar.gz		
Brazilian Portuguese	html.gz	tar.gz	chm	
French	html.gz	tar.gz	chm	
German	html.gz	tar.gz	chm	
Italian	html.gz	tar.gz	chm	
Japanese	html.gz	tar.gz	chm	
Polish	html.gz	tar.gz	chm	
Romanian	html.gz	tar.gz	chm	
Russian	html.gz	tar.gz	chm	
Spanish	html.gz	tar.gz	chm	

图 1-28 PHP 帮助文档下载页面

1.3　PHP 配置指令介绍

修改 PHP 配置指令可以在 PHP 页面中增加一些新的功能，例如限制使用文件下载功能。指令的修改基本上都是在 Apache 的 httpd.conf 和 PHP 的 php.ini 文件中进行，这两个文件中包含了大量的指令，这些指令控制了 Apache 和 PHP 的行为，并且决定了 PHP 在运行时出现的形式，本节将详细介绍 PHP 中常用的配置指令。

1.3.1　管理 PHP 的配置指令

配置指令的使用是为了使 PHP 功能更加强大，在 PHP 技术中有下面几种方法可以设置配置指令的值，分别为修改 php.ini、httpd.conf 和.htaccess 文件，也可以直接通过 PHP 脚本处理。

1. php.ini

PHP 有两个配置模板：php.ini-dist 和 php.ini-recommended，建议使用后者，因为其中的许多参数都已经设置为推荐值。如果采纳这个建议，在保证安装安全以及调整安装时，就能节省大量的时间和精力，因为这个文件中有大约 240 个不同的配置参数，默认值有助于快速地部署 PHP。如果还想对 PHP 的行为做另外的调整，那么就需要对这个文件有所了解，学习其中的配置参数。

与 Apache 的 httpd.conf 文件或者 MySQL 的 my.cnf（Windows 下是 my.ini）类似，php.ini 文件是 PHP 的全局配置文件。这个文件处理了 PHP 在 12 个不同方面的行为。

- 语言选项；
- 安全模式；
- 语法突出显示；
- 杂项；
- 资源限制；
- 错误处理和日志；
- 数据处理；
- 路径和目录；
- 文件上传；
- Fopen 包装器；
- 动态扩展；
- 模块设置。

php.ini 文件是一个纯文本文件，只包含注释和“参数=值”赋值对。代码 1.3 为 php.ini 文件中的一个示例片段。

代码 1.3　php.ini 文件示例代码片段

```
; Note: Never use this feature for production boxes.
```

```
;docref_root = "/phpmanual/"
;docref_ext = .html
; String to output before an error message.
;error_prepend_string = "<font color=#ff0000>"
; String to output after an error message.
;error_append_string = "</font>"
; Log errors to specified file.
;error_log = filename
; Log errors to syslog (Event Log on NT, not valid in Windows 95).
;error_log = syslog
```

在代码 1.3 中，以分号开头的行表示已经注释的配置或说明。

如果很清楚某个配置参数的作用，可以考虑将其注释删除，使文件的内容简化，从而减少以后的编辑时间。

修改过的配置文件何时生效，取决于安装 PHP 的方式。如果安装 PHP 作为 CGI 二进制包，那么每次调用 PHP 时都会重新读取 php.ini 文件，因此修改将立即生效。如果安装 PHP 作为 Apache 模块，则只会在 Apache 守护进程第一次启动时读取 php.ini。因此，如果以后一种方式安装 PHP，就必须重启 Apache，这样修改才会生效。

2. Apache 的 httpd.conf 和.htaccess 文件

当 PHP 作为 Apache 模块运行时，就可以通过 httpd.conf 或.htaccess 文件修改指令。为此，可以在“名=值”对前面加上以下某个关键字作为前缀，详细信息如表 1-1 所示。

表 1-1　指令信息表

指令格式	说明
php_value	设置指令的值
php_flag	设置指定布尔指令的值
php_admin_value	设置指定指令的值，它与 php_value 不同，不能用在 htaccess 文件中，也不能在虚拟主机或.htaccess 中被覆盖
php_admin_flag	设置指定布尔指令的值，它与 php_flag 不同，不能用在.htaccess 文件中，也不能在虚拟主机或.htaccess 中被覆盖

3. 执行脚本

第三种处理 PHP 配置变量的方式是通过 ini_set()方法来完成，这也是最本地化（localized）的方式。假设要修改 PHP 中给定脚本的最大执行时间，只需在脚本最上面加入如下命令。

```
ini_set("max_execution_time","60" );
```

4. 配置指令作用域

并不是任何地方都可以修改配置指令，因为当修改配置指令时，可能会引起安全隐患。这是由配置指令作用域不同而引起的，每个指令都有自己的作用域，指令只能在其作用域中

修改，总共有 4 个不同作用域，详细信息如表 1-2 所示。

表 1-2　指令作用域

指令名称	作用域说明
PHP_INI_PERDIR	指令可以在 php.ini、httpd.conf 或.htaccess 文件中修改
PHP_INI_SYSTEM	指令可以在 php.ini 和 httpd.conf 文件中修改
PHP_INI_USER	指令可以在用户脚本中修改
PHP_INI_ALL	指令可以在任何地方修改

1.3.2　PHP 的配置指令

上面小节介绍了怎样配置和管理 PHP 的配置指令，本节将对常用的配置指令进行说明。

1．语言选项

该部分指令主要用于确定语言最基本的一些行为，其配置的详细信息如表 1-3 所示。

表 1-3　语言选项信息

指令	作用域	说明
engine	PHP_INI_ALL 默认值为 on	确定 PHP 引擎是否可用
zend.zel_compatibility_mode()	PHP_INI_ALL 默认值为 on	PHP 5.0 和 PHP 4.0 之间存在不兼容的特性。如果启动该指令，可以在 PHP 5.0 中运行 PHP 4.0 的程序
short_open_tag	PHP_INI_ALL 默认值为 on	可以在 PHP 文件中使用段标记<??>界定 PHP 代码。如果要和 XML 结合使用 PHP，可以禁用此选项以便于嵌入使用<?xml ?>
asp_tags	PHP_INI_ALL 默认值为 off	用来设置是否支持 ASP 风格的界定符
precision integer	PHP_INI_ALL 默认值为 12	设置浮点数中显示的有效数字个数
y2k_compliance(on,off)	PHP_INI_ALL 默认值为 off	禁用 y2k_compliance 参数
output_buffering(on,off)	PHP_INI_ALL 默认值为 off	用来设定是否使用缓冲，启用 output_buffering 指令将打开缓冲
output_handler	PHP_INI_ALL 默认值为 Null	用来设置在把输出返回给请求用户之前，将输出传递给一个函数
allow_call_time_pass_reference（on,off）	PHP_INI_SYSTEM 默认值为 on	函数有两种传值方式，按值和引用。此指令可以在函数中指定每个参数在函数调用时如何传递
serialize_precision(integer)	PHP_INI_ALL 默认值为 100	确定在串行化双精度和单精度浮点数时小数点后存储的位数
implicit_flush	PHP_INI_SYSTEM 默认值为 off	启用该命令后，每次调用 print 或 echo 后，以及每个嵌入的 HTML 块完成后，将会自动刷新或清除其内容的输出缓冲区

2．资源限制

PHP 5.0 在处理资源功能方面有很大的进步，但在执行程序时还要确保 PHP 脚本不会因为程序员或者用户的动作而独占服务器资源。有 3 个方面可能会过度耗费资源，分别为脚本执行时间、脚本输入处理时间和内存。资源消耗可以使用下面 3 个指令来控制，其详细信息如表 1-4 所示。

表 1-4　资源限制指令

指令	作用域	说明
max_execution_time(integer)	PHP_INI_ALL 默认值为 30	设置 PHP 脚本执行时间的上限，以秒为单位。如果设置为 0，将取消最大限制
max_input_time(integer)	PHP_INI_ALL 默认值为 60	设置 PHP 脚本解析请求数据所用的时间，以秒为单位。在文件上传时，该参数特别有用
memory_limit(integer)	PHP_INI_ALL 默认值为 8MB	设定了一个脚本所能够申请到的最大内存字节数。这有助于防止写得不好的脚本消耗光服务器上的可用内存。要使用此指令必须在编译的时候激活

3．安全模式

在多用户环境中部署 PHP 时，可能要限制 PHP 的功能。因为在多用户环境中，如果为每个用户都提供 PHP 的所有功能，就会暴露服务器的漏洞，还可能会破坏服务器的资源和文件。所以 PHP 应采用一种受限模式或安全模式运行。但启用安全模式会有很多影响，包括自动禁用很多功能和可能不安全的各种特性。可以使用函数进行限制，也可以通过在 PHP 中配置指令进行限制，其指令信息如表 1-5 所示。

表 1-5　安全模式指令

指令	作用域	说明
safe_mode(on,off)	PHP_INI_SYSTEM 默认值为 off	启用该指令表示 PHP 在上述约束条件下运行
safe_mode_gid(on,off)	PHP_INI_SYSTEM 默认值为 off	启用安全模式时，如果又启用了 safe_mode_gid(on,off)，在打开文件时，就会强制完成 UID 检查
safe_mode_include_dir(string)	PHP_INI_SYSTEM 默认值为 Null	启用该指令时，会使上面两个指令在指定位置失效，即打开指定文件夹时，将忽略 UID/GID 检查
safe_mode_exec_dir(string)	PHP_INI_SYSTEM 默认值为 Null	启用该指令，会限制通过 exec()函数只能执行指定目录中的可执行程序
safe_mode_allowed_env_vars(string)	PHP_INI_SYSTEM 默认值为 PHP_	启用该指令会限制用户能通过 PHP 脚本修改操作系统的变量
safe_mode_protected_env_vars(string)	PHP_INI_SYSTEM 默认值为 LD_LIBRARY_PATH	可以明确防止修改某些环境变量

续表

指令	作用域	说明
open_basedir（string）	PHP_INI_SYSTEM 默认值为 Null	启用该指令可以建立一个基目录，所有文件操作都限制在此目录中
disable_functions(string)	PHP_INI_SYSTEM 默认值为 Null	启用该指令，可以禁用某些函数
disable_classes(string)	PHP_INI_SYSTEM 默认值为 Null	启用该指令可以禁用某些类
ignore_user_abort(off,on)	PHP_INI_ALL 默认值为 on	启用该指令，可以使服务器忽略由于用户或浏览器引起的中断所造成的会话中止

4．数据处理

外部变量就是通过一些外部源传递给脚本的变量，例如 GET、POST、cookie、操作系统和服务器提供的外部数据。可以使用指令来影响外部变量的使用，这些指令详细信息如表 1-6 所示。

表 1-6　数据处理指令

指令	作用域	说明
register_globals(on,off)	PHP_INI_SYSTEM 默认值为 off	设置 ENVIRONMENT、GET、POST、COOLE、SERVER 变量为全局变量
register_long_arrays(on,off)	PHP_INI_SYSTEM 默认值为 off	该指令确定是否继续使用已经废弃的语法
register_argc_argv(on,off)	PHP_INI_SYSTEM 默认值为 on	设定通过 GET 方法传入变量信息与可执行文件传递参数类似
post_max_size(integer)	PHP_INI_SYSTEM 默认值为 8MB	设定 PHP 脚本以 POST 方法传递数据量的值
magic_quotes_gpc(on,off)	PHP_INI_SYSTEM 默认值为 on	确定是否对 GET、POST 和 cookie 方法传输的数据启用魔法引号
magic_quotes_sybase(on,off)	PHP_INI_ALL 默认值为 off	此参数只在启用 magic_quotes_runtime 时有效，如果启用了 magic_quotes_sybase，所有来自外部资源的数据都将使用一个单引号而不是反斜线进行转义
auto_prepend_file(string)	PHP_INI_SYSTEM 默认值为 Null	设置在 PHP 文件中要加载的文件名和相应路径
default_mimetype(string)	PHP_INI_ALL 默认值为 SAPI_DEFAULT_MIMETYPE	设置 PHP 文件的类型
default_charset(string)	PHP_INI_ALL 默认值为 SAPI_ DEFAULT_MIMETYPE	设置 PHP 文件在 Content-type 首部中输出字符编码形式。默认情况下是 iso-8859-1
variables_order (string)	PHP_INI_ALL 默认值为 Null	该指令确定 ENVIRONMENT、GET、POST、COOLE、SERVER 变量的解析顺序
arg_separator.input (string)	PHP_INI_ALL 默认值为&	&是 POST 或 GET 方法用来分隔输入变量的标准字符

续表

指令	作用域	说明
arg_separator.output (string)	PHP_INI_ALL 默认值为&	能自动生成 URL，并使用标准的&符号分隔输入变量

5. 路径和目录

路径和目录指令主要用来设置 PHP 文件的默认路径，这些路径用于导入函数库和扩展包以及确定用户 Web 目录和 Web 文档目录。其指令的详细信息如表 1-7 所示。

表 1-7　路径和目录指令

指令	作用域	说明
include_path (string)	PHP_INI_ALL 默认值为 Null	设定 include()、require()和 fopen_with_path()等函数使用的基本路径，可以指定多个目录
doc_root	PHP_INI_SYSTEM 默认值为 Null	此参数提供所有 PHP 脚本的默认位置，此参数非空时才会用
user_dir(string)	PHP_INI_SYSTEM 默认值为 Null	用来指定在使用/~username 约定打开文件时，PHP 使用的绝对目录
extension_dir(string)	PHP_INI_SYSTEM 默认值为 PHP_EXTENSION_DIR	设置 PHP 可加载的扩展包的位置
enable_dl(on,off)	PHP_INI_SYSTEM 默认值为 on	允许用户在运行时加载 PHP 扩展包

include_path 指令是一个比较重要的指令，在不同的系统中具有不同的写法。格式和系统的 PATH 环境变量类似，例如在 UNIX 下用冒号分隔，在 Windows 下用分号分隔，如代码 1.4 所示。

代码 1.4　include_path 指令写法

```
例 H.1. UNIX include_path
include_path=".:/php/includes"
例 H.2. Windows include_path
include_path=".;F:\php\includes"
```

6. 文件上传

PHP 文件支持 POST 方法上传和管理文本文件以及二进制文件。有 3 个指令支持这个功能，如表 1-8 所示。

表 1-8　文件上传指令

指令	作用域	说明
file_uploads (on,off)	PHP_INI_SYSTEM 默认值为 on	是否允许 HTTP 文件上传
upload_tmp_dir(string)	PHP_INI_SYSTEM 默认值为 Null	设定文件上传时存放文件的临时目录
upload_max_filesize(integer)	PHP_INI_SYSTEM 默认值为 2MB	所上传的文件的最大字节数

1.4　PHP 简单例子

本节将创建一个 PHP 的简单示例，让读者简单了解 PHP，该示例主要是显示一个字符串信息。在前面的章节中安装了 PHP 运行所需要的软件，首先新建一个文本文件，然后输入如代码 1.5 所示的代码。

代码 1.5　示例代码

```
<html>
  <head>
    <title>演示示例</title>
  </head>
  <body>
    <h1>PHP 简单示例</h1><hr/>
    <?php
        echo ("Hello World");
    ?>
  </body>
</html>
```

输入代码后，将文件保存到 F:\MyWeb\Apache\htdocs 目录下，并重命名为 demo.php。然后打开 IE 浏览器，在地址栏中输入 http://localhost/demo.php，单击【转到】按钮，页面效果如图 1-29 所示。



图 1-29　页面效果图

在代码 1.5 中，可以看出 PHP 代码可以嵌入 HTML 标记语言中，嵌入的方法是使用<?php ?>。在嵌入标记中，使用输出函数输出了一个字符串信息。如果开发一个大型 PHP 项目，不可能将代码都放到 htdocs 文件夹内，而是需要建立该项目的文件夹。如果要建立一个 myweb 项目，可以直接在 F:\MyWeb\Apache\htdocs 目录下建立目录 myweb，将该项目的 PHP 文件放到里面即可。如果要建立的项目不想放到 F:\MyWeb\Apache\htdocs 目录的下面，而是在另外的盘符下建立一个放项目的目录，如 E 盘的 Test 目录，并在该目录下放置一个 info.php 页面。这时需要打开 F:\MyWeb\Apache\conf 目录下的 httpd.conf 文件，在<Directory "F:/Web/apache/htdocs">标记的上面添加如代码 1.6 所示的代码，输入完成后保存 httpd.conf 即可。

代码 1.6　httpd.conf 文件配置代码

```
Alias /Test " E:/Test "
<Directory "E:/Test">
  Options Indexes FollowSymLinks
```

```
    AllowOverride None
    Order allow,deny
    Allow from all
  </Directory>
```

下面在 E:/Test 目录下创建一个 info.php 文件，并在该文件中输入 PHP 代码，然后保存。打开 IE 浏览器，在地址栏中输入 http://localhost/Test/info.php，单击【转到】按钮，页面会显示服务器相关信息，如图 1-30 所示。



图 1-30　Test 文件及页面效果

第2章 PHP基础语法



PHP 是一种易于使用的服务器端脚本语言，只需要很少的编程知识就能使用 PHP 建立一个真正具有交互功能的 Web 站点。PHP 与微软的 ASP 有几分相似，都是在服务器端执行并嵌入到 HTML 文档的脚本语言，但是 PHP 独特且创新的语法，更容易被初学者掌握。本章将对 PHP 的基础语法进行详细说明和介绍。


学习目标 Objective

- 掌握如何将 PHP 代码嵌入到 HTML 中
- 灵活应用 PHP 注释
- 掌握 PHP 的常用数据类型以及数据类型之间的转换
- 掌握如何创建 PHP 变量
- 掌握 PHP 表达式以及操作数和操作符
- 熟练运用 PHP 控制语句
- 熟练运用 PHP 内置函数
- 掌握如何自定义函数
- 熟练创建数组和掌握管理数组函数

2.1 PHP 脚本基础

本节介绍 PHP 脚本的一些基础知识，主要是在 HTML 文件中如何嵌入 PHP 脚本、应用注释以及输出服务器端信息等。

2.1.1 嵌入 PHP 代码

通过第一章的学习，读者应该了解 PHP 是一种可以直接嵌入到 HTML 语言中的脚本语言。将嵌入在 HTML 文档中的 PHP 脚本与其他语言分开，可以避免 PHP 引擎解释器将每一行代码都当作 PHP 脚本语言解释。本节主要讲解如何将 PHP 脚本与其他语言界定开，这样解释器就可以只解释 PHP 脚本语言，从而提高程序运行速度。

要将 PHP 脚本语言与其他语言分开，就要将嵌入的 PHP 脚本语言放置到特定的、成对的

标记内。这样，当解析一个 PHP 文件时，会寻找相应的开始和结束标记，这些标记告诉 PHP 解析器开始和停止的位置。此种解析方式可以使 PHP 嵌入到各种不同的文档中，在一对开始和结束标记之外的内容中会被 PHP 解析器忽略。大多数情况下 PHP 都是嵌入在 HTML 文档中。在 HTML 页面嵌入 PHP 脚本，常用的有 4 种方式，下面分别介绍。

1. 默认语法

默认标记包括开始标记“<?php”和结束标记“?>”，如下代码所示，程序运行时，PHP 解析器将只解析开始标记“<?php”和结束标记“?>”之间的程序代码，即“echo '使用默认标记';”。

```
<h1><a herf="http://www.itzcn.com">欢迎光临窗内网</h1>
<?php
echo '使用默认标记';
?>
```

2. 脚本标记

PHP 还支持一种主流的语言界定形式，即此处介绍的脚本标记。脚本标记是以“<script language="php">”开始，以“</script>”标记结束，脚本标记具体应用如代码 2.1 所示。将该文件保存为 1.php，运行结果如图 2-1 所示。

代码 2.1　使用脚本形式向 HTML 文档嵌入 PHP

```
<h1><a herf="http://www.itzcn.com">欢迎光临窗内网</h1>
<script language ="php">
    echo "使用脚本标记同样可以把 php 语言与其他语言界定开";
</script>
```

3. 短标记

短标记是一种不太常见的形式，以“<?”开始，以“?>”结束。使用这种更短的界定语法将 PHP 语言与其他标记界定开，短标记必须声明默认语法中必须使用的 php 引用。使用这个特性，必须启用 PHP 的 short_open_tag 指令。启用短标记，在使用 print 或 echo 显示信息时，还可以省略这些语句，短标记具体应用如代码 2.2 所示。

代码 2.2　使用短标记嵌入 PHP 脚本

```
<?= "使用短标记，输出信息"?>
<?
    echo "使用短标记输出信息";
?>
```

4. ASP 风格

ASP 风格标记是以“<%”开始，以“%>”结束，该风格标记和 ASP、JSP 中嵌入脚本程

序的标记一样，但是在 PHP 中必须设置 php.ini 文件。打开 F:\MyWeb\PHP 文件夹下的 php.ini 文件，该文件的 133 行如代码 2.3 所示，在这里将 asp_tags 属性的值设置为 On，该标记即可直接使用。

代码 2.3　ASP 风格标记

```
; Allow ASP-style <% %> tags.
asp_tags = Off
```

本节所讲解的这些标记，还可以在同一个页面中混合使用，这种用法具体如代码 2.4 所示。

代码 2.4　使用多种代码块

```
<html>
<head>
<title>多种代码块的使用</title>
<!--使用 JavaScript 脚本调用 PHP 页面-->
<script language="JavaScript">
      window.open("1.php");
</script>
</head>
<body>
    <h1>多种代码块的使用</h1>
    <?php
        $string_value="hello world";
     ?>
     <font size=6 color=red><%=$string_value%></font>
</body>
</html>
```

保存上述代码，在浏览器中打开该文件，单击【转到】按钮，将显示如图 2-1 所示的效果。

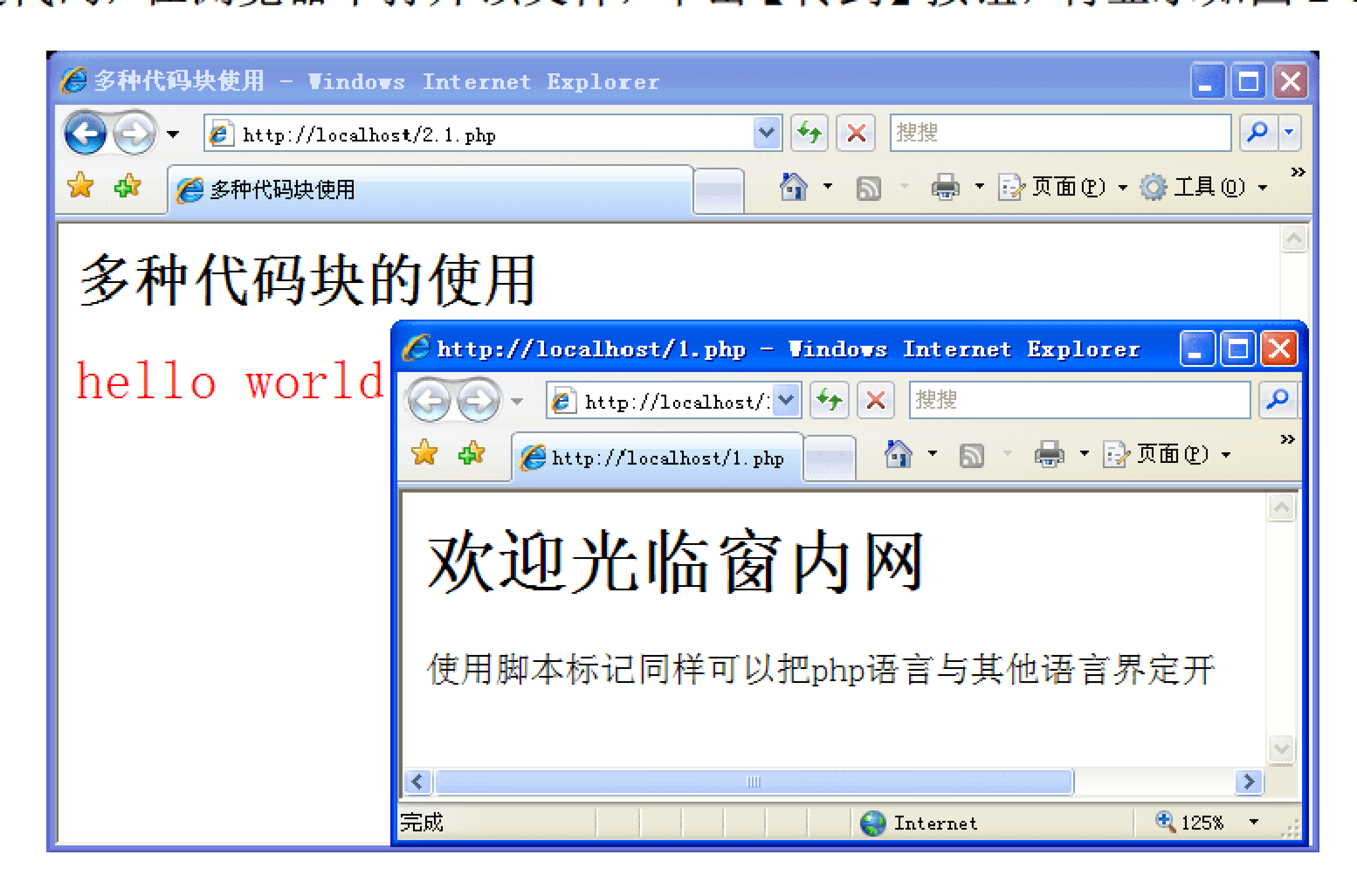


图 2-1　使用多种代码

2.1.2 注释

对于程序开发初学者来说，务必要养成在代码中添加注释的良好习惯，一来方便日后自己调试代码，另一方面也方便他人阅读，特别是多人协同开发项目时。在 PHP 中支持多种注释方式，例如单行注释、多行注释和 C 语言风格注释等，分别介绍如下。

1. 单行注释

在 PHP 中添加注释时，如果注释的内容没有超过一行，那么就可以使用单行注释，因为这种注释很短，没有必要界定这种注释的结束。这种注释和 C++中的单行注释相同，因此也可以称为单行 C++注释。具体用法如下所示：

```
<?php
   //在页面上输出信息
   echo  "Hello World~! ";
?>
```

2. Perl 风格单行注释

Perl 风格单行注释以#开头，该风格同样也是 C++风格，可以将上面的示例重写如下所示：

```
<?php
     #使用 Perl 风格单行注释  在页面上输出信息
     echo  "Hello World~!";
?>
```

3. 多行注释语法

多行注释应用于注释内容较多需要换行的情况，以“/*”标记开始，以“*/”结束，用法如下所示：

```
<?php
   /*
     该段代码主要是获取用户信息
     例如用户名、密码
     获取这些内容之后再向数据库验证，验证无误后让用户登录系统
*/
?>
```

在运行以上程序代码时，PHP 解析器将直接跳过“/*”和“*/”之间的内容，执行 PHP 程序代码。

2.1.3 输出函数介绍

使用 PHP 技术可以使应用程序具有高度的交互性，从而使客户端与服务器端很好地交互

信息。在 PHP 中从服务器端获得信息，如果要输出到页面上，通常使用一些 PHP 内置的函数，例如 echo()、print()、printf()和 sprintf()函数。本节主要介绍这些函数的作用，语法格式，以及如何运用这些函数。

1．print()函数

该函数主要负责为用户提供反馈，能显示原始字符串和变量，如果输出成功，就会返回一个布尔值，该函数语法格式如下所示：

```
boolean print(argument)
```

由 print()函数语法格式知，该函数只有一个参数，并且返回的是布尔值。下面创建一个示例，具体讲解如何使用该函数，代码如下所示：

```
<?php
print "<p> 我喜欢使用 PHP</p>";
?>
<?php
$str="使用 PHP";
print("<p> 我喜欢.$str</p>");
?>
```

在上述代码中，创建一个变量，然后使用“.”连接字符串与变量，这里的圆点的含义是字符串连接符。执行该段代码，结果如下所示：

```
我喜欢使用 PHP
我喜欢使用 PHP
```

2．echo()函数

echo()语句与 print()功能相同，都是用来向客户端输出信息。但有两点不同，echo()语句不能用在复杂表达式中，因为它返回 void，而 print()返回一个 boolean 值。其次，echo()能输出多个字符串。虽然 echo()和 print()在功能上可以互换，但是在执行速度上 echo()稍快一些。该函数语法格式如下所示：

```
void echo ( string $arg1 [, string $...] )
```

由于该函数和 print()函数功能相同，在此就不再列举示例。

3．printf()函数

该函数与前面讲解的两个函数功能相同，都是向客户端输出字符串，但是使用 printf()函数输出的字符串是一个格式化过的字符串，该函数语法格式如下所示：

```
int printf ( string $format [, mixed $args [, mixed $...]] )
```

由该函数的语法格式可以看出，函数的返回值是一个整型数值，该整数表示字符串的长度。args 表示指定的参数，但是它的输出将根据 format 进行格式化。format 参数可以对输出

数据进行充分的控制，如对齐方式、精度、类型或位置。这个参数由 5 部分组成，如表 2-1 所示。

表 2-1　参数名称

参数位置	参数名称	参数说明
第 1 个	填充提示符	可选，这一部分确定为达到正确的字符串大小所用的填充字符。默认为空格，也可以指定其他填充提示符（在字符前加一个单引号）
第 2 个	对齐提示符	可选，这一部分确定输出是左对齐还是右对齐。默认为右对齐，可以用一个负号设置为左对齐
第 3 个	宽度提示符	可选，这一部分确定此函数输出的最少字符数
第 4 个	精度提示符	可选，确定应显示的小数位数，这一部分只影响浮点数类型的数据
第 5 个	类型提示符	这一部分确定如何转换参数

所支持的类型提示符如表 2-2 所示。

表 2-2　格式化类型表

类型	描述
%b	将参数认为是一个整数，显示为二进制数
%c	将参数认为是一个整数，显示为对应的 ASCII 字符
%d	将参数认为是一个整数，显示为有符号十进制数
%f	将参数认为是一个浮点数，显示为浮点数
%o	将参数认为是一个整数，显示为八进制数
%s	将参数认为是一个字符串，显示为字符串
%u	将参数认为是一个整数，显示为无符号十进制数
%x	将参数认为是一个整数，显示为小写的十六进制数
%X	将参数认为是一个整数，显示为大写的十六进制数

下面创建一个示例，具体讲解如何运用该函数，如代码 2.5 所示。

代码 2.5　printf()函数的使用

```
<?php
  printf("(1.1*10)=%d",(1.1*10));
  print "<br/>";
  $hh=8.15;
  printf("%.3f",$hh);
  print "<br/>";
  printf("%1\$s %2\$s, %2\$s","动物","海岸");
?>
```

保存并执行代码 2.5，结果如下所示：

```
(1.1*10)=11
8.150
动物 海岸, 海岸
```

4．sprintf()函数

sprintf()函数和 printf()函数一样，都是输出格式化后的信息，由于该函数与 printf()函数功能基本相同，在此就不再赘述，该函数语法格式如下所示：

```
string sprintf ( string $format [, mixed $args [, mixed $...]] )
```

下面使用该函数创建一个示例，具体讲解如何使用 sprintf()函数，具体代码如下所示：

```
<?php
$str=sprintf("%01.2f",43.2);
echo($str);
?>
```

2.2　数据类型

数据类型是具有相同特性的一类数据的统称。在 PHP 中，常见的数据类型包括字符串、整型、浮点型和布尔型等，并且 PHP 还在一直不断地添加新的数据类型，表 2-3 列出了 PHP 中常见数据类型。

表 2-3　数据类型表

数据类型	数据类型名称	类别	数据类型	数据类型名称	类别
boolean	布尔型	标量数据类型	array	数组	复合数据类型
integer	整型	标量数据类型	object	对象	复合数据类型
float	浮点型	标量数据类型	resource	资源	特殊数据类型
string	字符串	标量数据类型	NULL	空型	特殊数据类型

以上数据类型可以分为 3 类，分别为：标量数据类型、复合数据类型和特殊数据类型。

2.2.1　标量数据类型

标量数据类型只能包含单一数据信息。例如布尔型数据、整型数据、浮点型数据和字符串型数据等。本节将详细介绍标量数据类型的使用方法。

1．布尔型数据类型

布尔数据类型表示真实性，只支持两个值，可以为 true 或 false，这两个值不区分大小写。也可以用 0 表示 false，非 0 值表示 true。布尔数据类型使用示例如下所示：

```
$a = false;
$a = 0;          //表示 false
$a = -1;         //表示 true
$a=5;            //表示 true
```

2. 整型数据类型

整数就是一个不包含小数部分的数，整数用十进制、十六进制或八进制指定，并且前面可以加上“+”号或者“–”号。如果用八进制符号，数字前必须加上 0，用十六进制符号，数字前必须加上 0x。



PHP 所能支持的最大整数与平台有关，一般是 $\pm 2^{31}$，如果在 PHP 中超过了此限制，将自动转换为浮点数。

3. 浮点数据类型

浮点数（floating-point number）也称为单精度数（float）、双精度数（double）或实数（real number），可以指定包含小数部分的数。浮点数用于表示货币值、重量、距离，以及用简单的整数无法满足要求的其他表示。

4. 字符串数据类型

字符串表示一个字符序列，也可以看作一个连续数组，并且这样的序列或者数组通常用单引号或双引号界定。另外还有一种创建字符串的方法是使用定界符“<<<”，首先在“<<<”之后提供一个标识符，然后是字符串，最后是同样的标识符结束字符串。结束标识符必须从行的第一列开始。同样，标识符也必须遵循 PHP 中其他任何标签的命名规则：只能包含字母、数字、下划线，而且必须以下划线或非数字字符开始。“<<<”符号表示一个定界符，EOD 表示一个标识符，标识符中间是字符串信息。下面使用定界符创建一个字符串，具体代码如下所示：

```
$str = <<<EOD
采用定界符创建字符串，
并且该字符串是一个多行字符串
EOD;
```

在字符串中要表示一个单引号，需要用反斜线（\）转义。如果在单引号之前或字符串结尾需要出现一个反斜线，需要使用两个反斜线表示。注意如果试图转义任何其他字符，反斜线本身也会被显示出来，所以通常不需要转义反斜线本身。使用单引号创建的字符串如下所示：

```
<?php
echo  '这是一个简单的字符串<br/>';
echo  '看这个字符串: "yound tang"<br/>';
echo  '请删除 C:\\*.*?文件';
?>
```

保存并执行上述代码，结果如下所示：

```
这是一个简单的字符串
```

```
看这个字符串："yound tang"
请删除 C:\*.*?文件
```

创建一个字符串，也可以用双引号将字符括起来。在这种字符串的格式中，可以使用更多的转义字符。常用的转义字符如表 2-4 所示。

表 2-4　转义字符表

序列	含义
\n	换行
\r	回车
\t	水平制表符
\\	反斜线
\$	美元符号
\"	双引号
\[0-7]{1,3}	此正则表达式序列匹配一个用八进制符号表示的字符
\x[0-9A-Fa-f]{1,2}	此正则表达式序列匹配一个用十六进制符号表示的字符

2.2.2　复合数据类型

复合数据类型就是将多种相同的类型信息组合在一起，例如数组（Array）和对象（Object），本节主要介绍数组和对象。

1. 数组

数组是将一系列类似的具有相同数据类型的项组合在一起，并按照特定的方式进行排列和引用，例如可以在一个数组中放置多个整数值。在 PHP 中，数组里面的值（values）有序地排列，数组里存放的值可以通过数组键名（keys）加上数组名称来获得。数组具体应用如下所示：

```
<?php
    $city[0] = "北京";
    $city[1] = "上海";
    $city[2] = "重庆";
    $city[3] = "天津";
?>
```

如上代码使用了数组数字索引，同样也可以使用值与值直接关联，如下代码展示了使用值与值直接关联的用法：

```
$city[beijing] = "北京";
$city[shanghai] = "上海";
$city[chongqing] = "重庆";
$city[tianjin] = "天津";
```

除了上述创建数组的方法外，还可以使用 array()结构创建数组，下面使用 array()结构创建数组，具体代码如下所示：

```
<?php
   $city = array("北京","上海","天津","重庆");
   $citys = array('beijing'=>"北京",
             'shanghai'=>"上海",
             'tianjin'=>"天津",
             'chongqing'=>"重庆");
?>
```

2．对象

PHP 支持面向对象编程。面向对象编程促进了清晰的模块设计，简化了调试和维护，并且有助于提高代码可重用性。类是面向对象程序设计的单元，包含变量、属性和方法的结构定义。类用关键字 class 定义，在此只是简单介绍类，将在后面的章节中进行详细介绍。

要创建一个对象首先要创建一个类，创建好类可以使用 new 关键字生成类的对象，对象可以使用“=>”访问属性、方法和其他类成员。下面创建一个使用对象的示例，如代码 2.6 所示。

代码 2.6　创建对象

```
<?php
    Class Person
    {
      var $name = "";
      function GetName()
      {
         if(is null($this->name))
         {
          return $this->name;
         }
         else
         {
          return "唐晓阳";
         }
      }
    }
    $p = new Person();
    echo "你的姓名是: ".$p->GetName();
?>
```

执行代码 2.6，结果如下所示：

```
你的姓名是: 唐晓阳
```

2.2.3　特殊数据类型

在 PHP 中，除了上面常用的数据类型外，还存在一些用在特殊方面的数据类型。这些特殊数据类型主要提供某种特殊用途，因此无法归入其他任何类型。特殊数据类型包括资源和空数据类型。

1．资源

PHP 通常用于与一些外部数据源交互，例如数据库、文件和网路流等。通常这种交互通过句柄完成，这些句柄将保持对资源的引用，直到通信结束，才撤销句柄。这些句柄就属于资源数据类型。

当不使用资源时，资源将作为垃圾被收集起来。资源类型的变量并不真正保存一个值，实际上只保存一个指针，指向所打开的资源类型的变量。在使用时，直接调用指针即可，如果输出其内容，将看到一个资源 ID 号的引用。

2．null

null 表示空，不表示空格，不表示零，它表示没有值。一个变量在下列几种情况下，会被认为是 null 值：被赋值为 null、没有被赋值和使用 unset()函数清除等。null 的数据类型只有一个值 null。下面创建几个表示 null 的变量，然后使用 is_null()函数测试一个变量是否为 null，具体如代码 2.7 所示。

代码 2.7　使用 null 值

```
<?php
$default = "Yound Tang";
$default = null;        //变量值丢失
$default = Null;        //变量值丢失
$default = NULL;        //变量值丢失
if(is_null($default))
{
    echo "该变量为空值";
}
?>
```

保存并执行代码 2.7，结果如下所示：

```
该变量为空值
```

2.2.4　类型转换

将一种数据类型转换成另一种数据类型，称为数据类型转换。类型转换主要包括类型强制转换和类型自动转换两种，本节将对这两种类型转换进行详细介绍。

1. 强制转换

在 PHP 中，将一个变量强制转换成与原类型不同的另一种数据类型，称为类型强制转换。实现数据类型强制转换其实很简单，直接在变量的前面加上要转换的数据类型即可，转换过的变量，就以新的类型进行计算。表 2-5 列出了转换操作符。

表 2-5　类型转换符

转换操作符	转换为	转换操作符	转换为
(array)	数组	(object)	对象
(bool)或(boolean)	布尔值	(real)或(double)或(float)	浮点数
(int)或(integer)	整数	(string)	字符串

下面创建一个实例，具体讲解数据类型强制转换，如代码 2.8 所示。

代码 2.8　类型强制转换

```
<?php
  $i=55;
  $ii=(double)$i;
  echo($ii);
  echo("<br/>");
  $a=125.23;
  $aa=(int)$a;
  print($aa);
  echo("<br/>");
  $s="<a href='http://www.itzcn.com'>窗内网</a>";
  $ss=(int)$s;
  $sss=(object)$s;
  print($ss);
  echo("<br/>");
  print $sss->scalar;
?>
```

保存并执行代码 2.8，页面效果如图 2-2 所示。

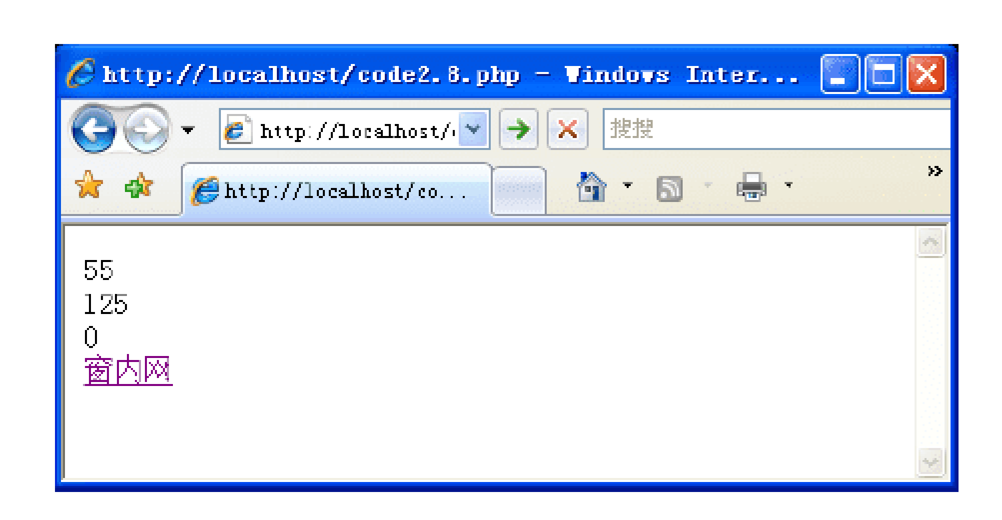


图 2-2　强制转换

从执行的结果可以看出，可以将一个整型转换为浮点型，也可以将一个浮点型转换为整型。当从数据类型范围大的向数据类型范围小的转换时，会舍弃一部分数值。例如在示例中，

从浮点型强制转换为整型时，舍弃了 0.23 小数位。当将一个字符串的类型转换为整型时，就会返回一个 0 值。任何一种数据类型都可以转换为对象，转换后，就成为该对象的一个属性，属性名为 scalar，可以通过对象名引用访问该属性。

2. 自动转换

由于 PHP 是一种弱类型的语言，所以 PHP 中的数据类型可以根据变量所处的环境来转换，将变量转换为最适合的类型，这就是本节要讲解的自动转换。下面创建一个实例，说明自动转换类型的使用情况，具体如代码 2.9 所示。

代码 2.9　类型自动转换

```
<?php
  $s1="10abc";
  $s2=10;
  $s1=$s2+$s1;
  echo($s1);
  echo("<br/>");
  $total="5.0";
  if($total)
      echo "\$total 自动转换为布尔值";
  echo("<br/>");
?>
```

保存并执行代码 2.9，结果如下所示：

```
20
$total 自动转换为布尔值
```

从执行结果中可以看出，如果整型变量$s2 和包含数值的字符串相加，该字符串会自动转换为整型变量相加。如果一个字符串的开始位置包含了数值，那么再和整型的变量相加，结果同样是一个整型值。$total 变量是一个整型变量，但是当用到条件语句中时，就会自动转换为布尔类型，其值为 true。

2.2.5　类型函数

类型函数由类型相关函数与类型标识符函数组成，本节将详细介绍这些函数。

1. 类型相关函数

PHP 是一种弱类型语言，在声明变量时，不带有数据类型。在 PHP 的操作中，需要获取某个变量的数据类型或者设置某个变量的数据类型时，需要使用 PHP 内置函数。PHP 中有两个获取变量数据类型的函数，settype()函数和 gettype()函数，下面详细介绍这两个函数。

- **settype()函数**　该函数将变量设置成指定的数据类型，语法格式如下所示：

```
bool settype ( mixed $var, string $type )
```

由语法格式知，该函数如果执行成功则返回 true，失败则返回 false。下面创建一个示例，具体讲解如何使用该函数，如代码 2.10 所示。

代码 2.10　使用 settype()函数

```
<?php
    $mystring="5 Name";
   $mybool =true;
   settype($mystring, "integer");
   echo $mystring."<br/>";
   settype($mybool, "string");
   echo $mybool."<br/>";
?>
```

执行代码 2.10，结果如下所示：

```
5
1
```

- **gettype()函数**　该函数主要用于获取指定变量的数据类型，并且返回 string 类型数据，返回值为变量的数据类型。该函数的语法格式如下所示：

```
string gettype ( mixed $var )
```

由语法格式知，该函数只有一个参数，表示指定要获取数据类型的变量。下面创建一个示例，如代码 2.11 所示。

代码 2.11　使用 gettype()函数

```
<?php
    $name ="唐晓阳";
    $sex = "男";
    $age =25;
    echo gettype($name)."<br/>";
    echo gettype($sex)."<br/>";
    echo gettype($age)."<br/>";
?>
```

执行代码 2.11，结果如下所示：

```
string
string
integer
```

2．类型标识符函数

在前面介绍了设置和获取某个变量的数据类型，本节主要讲解判断某个变量的数据类型的函数。这些函数主要包括 is_array()、is_bool()、is_float()、is_int()、is_null()、is_numeric()、

is_object()、is_resource()、is_scalar()和 is_string()。这些函数的语法格式相同，并且返回值全部是布尔值，在此仅以 is_int()为例讲解，该函数语法格式如下所示：

```
bool is_int ( mixed $var )
```

由语法格式知，该函数返回的是布尔类型值，如果为 int 类型则返回 true，否则返回 false，其他函数跟此函数意义基本相同。下面创建一个示例，具体讲解如何运用这些函数，如代码 2.12 所示。

代码 2.12　使用类型标识符函数

```
<?php
    $a = false;
    $b = 0;
    if (is_bool($a))
    {
        print "此变量为布尔值类型<br/>";
    }
    if (is_bool($b))
    {
        print "此变量为布尔值类型<br/>";
    }
    else
    {
        echo "此变量不为布尔值类型<br/>";
    }
?>
```

执行代码 2.12，结果如下所示：

```
此变量为布尔值类型
此变量不为布尔值类型
```

2.3　变量

变量是指在程序运行过程中随时可以发生变化的量。变量中可以存放字符串、数值、日期以及属性等。变量的值是临时的，当程序运行时，变量的值存在，程序一旦结束，变量的值就会丢失。本节将详细介绍如何创建变量，以及如何调用变量。

2.3.1　创建变量

PHP 中的变量使用美元符号$作为前缀，然后是变量名。$之后的部分为变量的标识符，标识符是一个标识不同对象的符号，如变量的名称，函数的名称，或者其他用户自定义对象

的名称。在 PHP 中，标识符的命名必须符合下面的规定。

- 标识符可以由一个或多个字符组成，但必须以字母或者下划线开头。此外，标识符只能由字母、数字、下划线和从 127 到 255 的其他 ASCII 字符组成。如 my_a、Ss、_value 这些标识符名称都是合法的，而 q^a、4tt 这些标识符名称是不合法的。
- 标识符区分大小写。因此，变量$recipe 不同于变量$Recipe、$rEciPe 或$recipE。
- 标识符可以是任意长度。这很有好处，因为这样一来，程序员就能通过标识符名准确地描述标识符的用途。
- 标识符名称不能与任何 PHP 预定义关键字相同。

在创建变量的过程中，先声明变量，再给变量赋值。由于 PHP 是一种弱类型语言，在声明变量时，不需要显式声明变量数据类型，变量可以存放任何类型的值。在 PHP 中，变量在运行时进行类型检查，并且可以用另一个不同类型的值取代变量的值，下面声明一个变量，并且让另一个不同类型的值取代变量的值，最后声明一个没有赋值的变量，具体代码如下所示：

```
$what = "Yound Tang";
$what = 25;
$name;
```

在 PHP 中，有两种方式可以为变量赋值，分别为：值赋值和引用赋值。值赋值是直接将一个数值通过赋值表达式为变量赋值，这种方式赋值会把该变量原来的数值覆盖；引用赋值表示所创建的变量与另一个变量引用的内容相同。在此首先讲解值赋值为变量赋值的方式。在声明变量时赋值是一种常用的变量的赋值方法，使用示例如下所示：

```
<?php
$name = "唐晓阳";
$age = "23";
$sex = "男";
echo "你的姓名是：".$name."<br/>";
echo "你的年龄是：".$age."<br/>";
echo "你的性别是：".$sex."<br/>";
?>
```

执行该段代码，结果如下所示：

```
你的姓名是：唐晓阳
你的年龄是：23
你的性别是：男
```

在 PHP 中，声明变量可以直接赋值，也可以不赋值。当需要使用变量存储值时，可以引用变量赋值，如果多个变量引用了同一个内容，修改其中任意一个变量，在其余的变量上都将有所反映。在等于号后面加一个&符号就可以完成引用赋值。引用赋值的示例形式如下所示：

```
$value1="Hello World";
$value2=& $value1;
$value2="GoodBye";
```

```
    echo $value1."<br/>";
    echo $value2."<br/>";
```

在上述代码中，创建一个变量 value1 并赋值为“Hello World”，在下面的语句中，变量 $value2 采用了引用赋值，即将 value1 的值赋给了 value2，此时这两个变量就是一个生命共同体了，当一个发生变化，另外一个就会显示出结果，该段代码执行结果如下所示：

```
GoodBye
GoodBye
```

2.3.2　变量作用域

声明变量的位置决定了变量的作用域，变量的作用域决定了程序中访问变量的范围。在 PHP 中，变量的作用域范围可以分为 4 类：局部变量、函数参数、全局变量和静态变量，下面分别介绍这 4 种作用域范围。

1．局部变量

在一个函数中声明一个变量是这个函数的局部变量，也就是说该变量只能被函数内部成员访问，函数外部成员不能访问该变量，并且不可见。下面创建一个使用局部变量的示例，如代码 2.13 所示。

代码 2.13　使用局部变量

```
<?php
    $count =10;
    function AddCount()
    {
        $count = 100;
        $count = $count + $count;
        echo $count;
        echo "<br/>";
    }
    AddCount();
    echo $count;
?>
```

执行代码 2.13，结果如下所示：

```
200
10
```

由输出结果知，该段代码输出了两个不同的值，这是因为函数 AddCount()中的变量为局部变量，修改局部变量的值不会影响函数外部的任何值，函数中的变量在 AddCount()函数执行结束时被抛弃，所以全局变量值还是 10。

2．函数参数

在 PHP 中，函数可以接受相应的参数，虽然这些参数是接受函数外部的值，但退出函数后就无法访问这些参数，参数的值也会消失。参数的值和函数的执行有很大的关系。函数参数是在函数后面的括号内声明，下面创建一个示例，如代码 2.14 所示。

代码 2.14　运用函数参数

```
<?php
   function EchoNum($age,$class)
   {
       echo "你的年龄是: ".$age."<br/>";
       echo "所在班级: ".$class;
   }
   EchoNum(18,"计算机技术与科学系 08 级 5 班");
?>
```

执行代码 2.14，结果如下所示：

```
你的年龄是: 18
所在班级: 计算机技术与科学系 08 级 5 班
```

函数参数也可以看作是局部变量，意味着这些参数只在函数内部起作用，在函数的外部不能访问这些变量。

3．全局变量

全局变量可以在整个 PHP 程序中任何地方访问。在函数中显式声明全局变量很简单，只需在函数中使用 global 关键字即可，下面创建一个使用全局变量的示例，如代码 2.15 所示。

代码 2.15　在函数中修改全局变量

```
<?php
   function AddNum()
   {
      global $num;
      $num = $num + $num ;
      echo $num ;
   }
   $num = 100;
   AddNum();
?>
```

执行代码 2.15，结果如下所示：

```
200
```

如果不在$num 前加 global，该变量会被认为是局部变量，此时页面上显示的值为 0。添

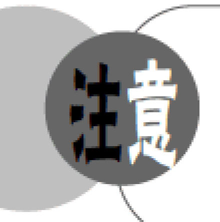
加 global 后，就可以修改全局变量了。声明全局变量还有另外一种方法，那就是使用 PHP 的$GLOBALS 数组。使用该数组和使用 global 的效果一样，下面创建一个使用$GLOBALS 数组的示例，如代码 2.16 所示。

代码 2.16　使用$GLOBALS 数组

```
<?php
   function AddNum()
   {
      $GLOBALS['num'] =$GLOBALS['num']+$GLOBALS['num']  ;
      echo "该值是: ".$GLOBALS['num'] ;
   }
   $num = 100;
   AddNum();
?>
```

执行代码 2.16，结果如下所示：

```
该值是: 200
```



在使用全局变量时一定要注意，因为一旦任何一个地方修改了全局变量的值，全局变量的值就发生了改变，这样很容易发生意外。

4. 静态变量

静态变量仅在局部函数域中声明，用关键字 static 可以声明一个静态变量。静态变量在函数退出时不会丢失值，并且再次调用此函数时还能保留值。下面创建一个使用静态变量的示例，如代码 2.17 所示。

代码 2.17　使用静态变量

```
<?php
   function keepNum()
   {
       static $num =0;
       $num ++;
       echo "现在静态变量的值是: ".$num;
       echo "<br/>";
   }
   $num = 10;
   echo "变量 num 的值是: ".$num."<br/>";
   keepNum();
   keepNum();
?>
```

执行代码 2.17，结果如下所示：

```
变量 num 的值是：10
现在静态变量的值是：1
现在静态变量的值是：2
```

由代码 2.17 执行结果知，由于在函数中指明了变量为静态变量，因此再执行函数时保留了前面的值。

2.3.3 变量的变量

创建一个变量，需要为该变量赋值，当然也可以引用赋值。有时也需要使用该变量的内容作为变量名，也就是说这个变量存放在另一个变量中，这称为变量的变量。定义变量的变量只需要在变量的前面再添加一个美元符号即可。下面创建一个示例，代码如下所示。

```
<?php
   $name = "Yound";
   $$name = " Tang";
   echo  $name.$$name;
?>
```

执行该段代码，结果如下所示：

```
Yound Tang
```

2.4 常量

常量是指在程序执行过程中无法修改的值，在程序中处理绝对不需要修改的值时，常量非常有用，例如在程序中定义圆周率。常量一旦定义，在程序的任何地方都不可以修改，并且可以在程序的任何地方访问常量。常量对大小写非常敏感，一般情况下常量标识符总是大写。

在 PHP 中使用 define()函数定义常量，该函数定义一个指定常量名的常量，并能够赋值，该函数的语法格式如下所示：

```
boolean define(string name,mixed value [, bool case_insensitive])
```

由该函数的语法格式可以看出，该函数有 3 个参数，其中 name 表示要定义的常量名称；value 表示常量的值；参数 case_insensitive 表示在引用该常量时，是否区分大小写，该值如果为 true，表示不区分大小写。下面创建一个示例，具体讲解如何使用该函数定义常量，代码如下所示：

```
<?php
    define("PI",3.1415927);
```

```
    echo  "该常量的值是: ".PI."<br/>";
    $pi = 2 * PI;
    echo "该常量的 2 倍是: ".$pi."<br/>";
?>
```


在上述代码中，定义了一个 PI 常量，并输出到页面上。执行该段代码，结果如下所示：

```
该常量的值是: 3.1415927
该常量的 2 倍是: 6.2831854
```

PHP 提供了大量的预定义常量，不过很多常量都是由不同的扩展库定义，只有在加载了这些扩展库时才能使用这些常量，或者动态加载，或者在编译时已经包括了这些扩展库。以下就是几个 PHP 预定义的常量，如表 2-6 所示。

表 2-6　常用常量

常量名	说明
__file__	当前 PHP 程序文件名
__line__	当前执行语句在 PHP 程序文件中的行数
PHP_version	当前 PHP 的版本号
PHP_os	当前所用操作系统类型
E_ERROR	指明最近一次产生不可恢复的错误
E_WARNING	指出有错误，但程序可以继续执行
E_PARSE	语法错误，分析器将被停止分析
E_NOTICE	产生异常，但不一定是错误

2.5　表达式

表达式是 PHP 语言的基础，表示程序中某个特定动作。所有的表达式至少由一个操作数和一个或多个操作符组成。本节主要讲解表达式的组成：操作数和操作符。

2.5.1　操作数

操作数就是在进行表达式计算时，需要使用的数值，最基本的表达式形式是常量和变量。下面创建一个操作数的示例，主要代码如下所示：

```
$a = 100;
$a++;
$sum = $a + $a;
```

2.5.2　操作符

操作符是表达式中指定某个动作的符号，例如两个数的相加、两个数的比较等。在 PHP

中，操作符两侧的操作数会自动进行类型转换，这在其他的编程语言中并不多见。常见的操作符种类很多，例如算术操作符、赋值操作符、逻辑操作符、比较操作符等。表 2-7 总结了 PHP 中的操作符。

表 2-7 PHP 中的操作符

结合方向	运算符	附加信息
非结合	New	创建新对象
右结合	[	数组下标
右结合	!	逻辑取反
右结合	~	逐位取反
右结合	++	自增
右结合	– –	自减
右结合	(int) (float) (string) (array) (object)	类型转换
右结合	@	不显示错误信息
左结合	* / %	算术操作符
左结合	+ – .	算术操作符和字符串操作符
左结合	<< >>	左移位、右移位
非结合	<,<= ,> ,>=	比较操作符
非结合	== != === !==	比较操作符
左结合	&	位操作符和引用
左结合	^	位操作符
左结合	|	位操作符
左结合	&&	逻辑操作符
左结合	||	逻辑操作符
左结合	? :	条件操作符
右结合	= += – = *= /= .= %= &= |= ^= <<= >>=	赋值操作符
左结合	and	逻辑操作符
左结合	xor	逻辑操作符
左结合	or	逻辑操作符
左结合	,	列表分隔符

通过对表 2-7 中操作符的学习，读者对这些操作符有了一定了解，下面详细介绍这些操作符的一些特性。

1. 操作符优先级

在一个表达式中操作符的求值顺序取决于操作符的相对优先级。操作符优先级是操作符的一个特性，确定以何种顺序计算周围的操作数，例如可以写出：

```
$count = 3 + 4 * 5;
```

该表达式可以等价地写为：

```
$count = 3+ ( 4 * 5 );
```

如上示例，加法操作符和乘法操作符具有不同的优先级，乘法的优先级要高于加法，所以乘法操作要比加法操作先进行。示例得到的正确结果应该是：3+20。如果优先级颠倒得到的结果将偏离正确结果很远，错误结果是：7*5。通过赋予操作数和操作符适当的顺序，相对优先级就能生成程序员想要的答案，这样就能够写出复杂的表达式。一个编写程序熟练的程序员，在处理复杂优先级规则时会将优先级规则缩减到两条：乘法和除法的优先级高于加法和减法；其他地方用括号。

2．操作符结合性

操作符的结合性指定了相同优先级运算的计算顺序。由表 2-7 知，操作符的结合性可以从左到右或者从右到左，从左到右结合性表示组成表达式的各种运算从左到右进行计算，从右到左则相反。下面创建一个示例，代码如下所示：

```
$value = 3*4/2*5;
```

该示例执行结果如下所示：

```
30
```

3．算术操作符

在程序语言中，求取两个数之间的和、差、积和商，通常采用算术操作符。PHP 语言中的算术操作符如表 2-8 所示。

表 2-8　算术操作符

例子	名称	结果	例子	名称	结果
-$a	取反	$a 的负值	$a*$b	乘法	$a 和$b 的积
$a+$b	加法	$a 和$b 的和	$a/$b	除法	$a 除以$b 的商
$a-$b	减法	$a 和$b 的差	$a%$b	取模	$a 除以$b 的余数

在上述的算术操作符中，除号（“/”）总是返回浮点数，即使两个运算数是整数（或由字符串转换成的整数）也是这样。取模$a%$b 在$a 为负值时的结果也是负值。

4．赋值操作符

所谓赋值操作符，完成的操作就是将一个数据值赋给一个变量。常用的赋值操作符如表 2-9 所示。

表 2-9　赋值操作符

例子	名称	结果	例子	名称	结果
$a=10	赋值	$a 等于 10	$a/=10	除法赋值	$a 等于$a 除以 10
$a+=10	加法赋值	$a 等于$a 加 10	$a.=10	拼接赋值	$a 等于$a 拼接 10
$a*=10	乘法赋值	$a 等于$a 乘以 10			

在表 2-9 中，$a=10 表示一个数据值赋值给变量$a，“=”是一个赋值符号，而不是一个等号。下面的 4 个赋值操作符，分别是二元操作符，其中$a+=10 表示的意思是$a=$a+10，其他

情况依次类推。

5. 比较操作符

比较操作符，顾名思义就是比较两个数值的大小，比较完成之后，返回的值为布尔值。比较表达式通常作为控制语句的判断条件。常见比较操作符如表 2-10 所示。

表 2-10　比较操作符

例子	名称	结果
$a==$b	等于	TRUE，如果$a 等于$b
$a===$b	全等	TRUE，如果$a 等于$b，并且它们的类型也相同
$a!=$b	不等	TRUE，如果$a 不等于$b
$a<>$b	不等	TRUE，如果$a 不等于$b
$a!==$b	非全等	TRUE，如果$a 不等于$b，或者它们的类型不同
$a<$b	小于	TRUE，如果$a 严格小于$b
$a>$b	大于	TRUE，如果$a 严格大于$b
$a<=$b	小于等于	TRUE，如果$a 小于或者等于$b
$a>=$b	大于等于	TRUE，如果$a 大于或者等于$b

在表 2-10 中列出了 PHP 中常用的操作符，下面创建一个示例，具体讲解这些操作符的运用，代码如下所示：

```
<?php
  $a=10;
  $b=10;
  if($a==$b)
  {
    echo"这两个值相等";
  }
?>
```

保存并执行上述代码，结果如下所示：

```
这两个值相等
```

6. 逻辑操作符

判断某一个对象是否符合标准，可能需要判断多个条件，这时就要用到逻辑操作符。逻辑操作符通常用在控制语句中，如 if 或 while 等。逻辑操作符使用的操作数是布尔类型的操作数。表 2-11 列出了常用逻辑操作符。

表 2-11　逻辑操作符

例子	名称	结果
$a and $b	And（逻辑与）	TRUE，如果$a 与$b 都为 TRUE
$a or $b	Or（逻辑或）	TRUE，如果$a 或$b 任一为 TRUE

续表

例子	名称	结果
$a xor $b	Xor（逻辑异或）	TRUE，如果$a 或$b 任一为 TRUE，但不同时为 TRUE
! $a	Not（逻辑非）	TRUE，如果$a 不为 TRUE
$a && $b	And（逻辑与）	TRUE，如果$a 与$b 都为 TRUE
$a || $b	Or（逻辑或）	TRUE，如果$a 或$b 任一为 TRUE

逻辑运算符使用示例如代码 2.18 所示。

代码 2.18　使用逻辑操作符

```
<?php
  $value=60;
  if($value>40 and $value<80)
    echo "温度过高，不适合洗澡";
  if($value>=35 and $value<=40)
    echo "该温度很适合洗澡";
?>
```

在代码 2.18 中，使用了逻辑操作符 and 判断两个条件是否都为 true 值，如果都为 true，则输出相应的语句。保存并执行代码 2.18，结果如下所示：

```
温度过高，不适合洗澡
```

7．字符串操作符

对于字符串操作的常用操作符有两个，第一个是连接运算符（“.”），它返回其左右参数连接后的字符串。第二个是连接赋值运算符（“.=”），它将右边参数附加到左边的参数后，如表 2-12 所示。

表 2-12　字符串连接符

示例	说明	输出
$a="唐"."晓阳"	拼接	$a 赋值为字符串“唐晓阳”
$a .="性别：男"	拼接赋值	$a 等于它的当前值与“性别：男”的拼接结果

8．自增和自减操作符

自增和自减操作符是一元操作符，可以使某一个变量自动加 1 或者减 1。常用的自增和自减操作符如表 2-13 所示。

表 2-13　自增和自减操作符

例子	名称	效果
++$a	前加	$a 的值加 1，然后返回$a
$a++	后加	返回$a，然后将 $a 的值加 1
--$a	前减	$a 的值减 1，　然后返回$a
$a--	后减	返回$a，然后将 $a 的值减 1

从表格中可以知道，操作符放置的位置不同，执行结果也不同。自增和自减操作符使用示例如代码 2.19 所示。

代码 2.19　运用自增自减操作符

```
<?php
echo "<h3>\$a++运算</h3>";
$a = 10;
echo "运算\$a++:  " . $a++ . "<br />\n";
echo "\$a 值" . $a . "<br />\n";
echo "<h3>++\$a 运算: </h3>";
echo "运算++\$a: " . ++$a . "<br />\n";
echo "\$a 值" . $a . "<br />\n";
?>
```

保存并执行代码 2.19，结果如下所示：

```
$a++运算
运算$a++:  10
$a 值 11
++$a 运算:
运算++$a: 12
$a 值 12
```

9．错误控制操作符

PHP 支持一个错误控制运算符：@。将其放置在一个 PHP 表达式之前，该表达式可能产生的任何错误信息都被忽略掉。@运算符只对表达式有效。对新手来说一个简单的规则就是：如果能从某处得到值，就能在其前面加上@运算符。

10．位操作符

位操作符在组成整数值的各个“位”层次上检查和处理整数值。位操作符如表 2-14 所示。

表 2-14　位操作符表

例子	名称	结果
$a & $b	And（按位与）	将把$a 和$b 中都为 1 的位设为 1
$a | $b	Or（按位或）	将把$a 或者$b 中为 1 的位设为 1
$a ^ $b	Xor（按位异或）	将把$a 和$b 中不同的位设为 1
~ $a	Not（按位非）	将$a 中为 0 的位设为 1，反之亦然
$a << $b	Shift left（左移）	将$a 中的位向左移动$b 次（每一次移动都表示“乘以 2”）
$a >> $b	Shift right（右移）	将$a 中的位向右移动$b 次（每一次移动都表示“除以 2”）

2.6　控制结构

PHP 支持多种传统编程结构来控制程序执行流程，例如 if 语句、for 循环语句和 while 循

环语句等。PHP 中的控制语句，可以分成 3 类：条件语句、循环语句和跳转语句。本节将详细介绍这些控制语句。

2.6.1　条件语句

条件语句又称为选择语句，判断一个表达式的结果是否满足条件。如果满足条件就执行该行语句，否则不执行该语句。在 PHP 中，条件语句有多种类型，分别为 if 语句、if…else 语句、if…else if…else 语句和 switch 语句等。本节主要讲解这些条件语句，详细介绍如下。

1. if…else 语句

在 PHP 中，if…else 语句的语法如下所示：

```
if (表达式)
{
    //表达式的结果为真时的运行程序段
}else
{
    //表达式的结果为假时的运行程序段
}
```

表达式结果值为布尔数据类型，如果布尔值为 true，就执行下面的大括号的语句块，否则会忽略语句。如果语句只有一条，这时可以将大括号省略。下面创建一个示例，具体讲解该条件语句的运用，如代码 2.20 所示。

代码 2.20　运用 if…else 条件语句

```
<?php
 $x = 10;
 $y = 15;
  if($x == $y)
  {
      echo "这两个值相同: ".$x;
  }
  else
  {
     echo "这两个值不相同; <br/>x 值是: ".$x."<br/>y 值是: ".$y;
  }
?>
```

代码 2.20 中，如果两个变量的值相同，则执行第一个语句，否则执行 else 语句。代码 2.20 执行结果如下所示：

```
这两个值不相同;
x 值是: 10
y 值是: 15
```

2. if…else if…else 语句

if…else 语句只能判断两种情况，要么符合条件，要么不符合条件。如果有多种情况，并且要求对每一种情况都做出反应时，if…else 语句就不能完成这样的任务。这时，可以使用 if…else if…else 语句来完成，也就是说可以多个 if…else 合并在一起使用，判断多条语句。if…else if…else 语句使用方式如下所示：

```
if(表达式 1)
{
    //表达式 1 为真时执行的程序段
}else if (表达式 2)
{
   //表达式 2 为真时执行的程序段
}else if (表达式 N)
{
   //表达式 N 为真时执行的程序段
}else
{
   //表达式 1…N 均为假时执行的程序段
}
```

下面使用该语句创建一个示例，详细讲解如何使用 if…else if…else 语句，如代码 2.21 所示。

代码 2.21　使用 if…else if…else 语句

```
<?php
 $x = 10;
 $y = 50;
 echo "当 x 为正数时，比较两个数的大小：<br/>";
 if($x < $y and $x>0)
 {
     echo "变量 x 小于 y,x 值：".$x."，y 值是：".$y."<br/>";
 }
 else if ($x == $y and $x >0)
 {
     echo "变量 x 等于 y,x 值：".$x."，y 值是：".$y."<br/>";
 }
 else if ($x > $y and $x>0)
 {
     echo "变量 x 大于 y,x 值：".$x."，y 值是：".$y."<br/>";
 }
 else
 {
     echo "变量 x 值小于或等于 0;x 值：".$x."y 值是：".$y;
 }
```

```
?>
```

执行代码 2.21，结果如下所示：

```
当 x 为正数时，比较两个数的大小:
变量 x 小于 y,x 值: 10，y 值是: 50
```


3．switch 语句

由示例 2.21 可以看到使用 if…else if…else 语句进行多个条件判断时的不足，尤其是当需要判断的条件越来越多时，书写非常麻烦，此时使用 PHP 提供的 switch 语句比较好。switch 语句语法如下：

```
switch (caseSwitch)
{
    case 1:
    echo("Case 1");
    break;
    case 2:
    echo("Case 2");
    break;
    default:
    echo("Default case");
    break;
}
```

代码 2.22 是 switch 多条件判断语句使用的示例。

代码 2.22　使用 switch 语句

```
<?php
$x = 70;
switch ($x)
{
    case 50:
        echo "你的成绩不及格";
        break;
    case 60:
        echo "你的成绩刚好及格";
        break;
    case 70:
       echo "你的成绩良好";
       break;
    case 80:
        echo "你的成绩优良";
        break;
    case 90:
        echo "你的成绩优秀";
```

```
        break;
}

?>
```

执行代码 2.22，结果如下所示：

```
你的成绩良好
```

2.6.2 循环语句

循环语句也称为迭代语句，让程序重复执行某个程序块，直到某个特定的条件表达式结果为假时，结束执行语句块。在 PHP 中常用的循环语句有下面几种形式： while 循环、do…while 循环、for 循环和 foreach 循环等，下面分别介绍这些循环语句。

1. for 循环语句

for 语句是一种在程序执行前就要判断条件表达式是否为真的循环语句。如果循环条件为假，那么循环语句根本就不会执行。for 循环适合用于一个语句或者语句块重复执行预定次数的循环，其语法如下所示：

```
for(初始值表达式; 循环条件表达式; 循环后的操作表达式)
{
    //执行语句块
}
```

下面创建一个示例，具体说明如何运用 for 循环语句，如代码 2.23 所示。

代码 2.23　使用 for 循环语句

```
<?php
  for ( $i = 1; $i < 5; $i++)
 {
     echo "第".$i."次执行该行代码<br/>";
 }
?>
```

执行代码 2.23，结果如下所示：

```
第 1 次执行该行代码
第 2 次执行该行代码
第 3 次执行该行代码
第 4 次执行该行代码
```

2. while 循环语句

while 循环语句是一种比较简单的循环。只要 while 语句中条件表达式的值为 true，就重

复执行循环语句。表达式的值在每次开始循环时检查，所以即使这个值在循环语句中改变了，语句也不会停止执行，直到本次循环结束。如果 while 表达式的值一开始就是 false，则循环语句一次都不会执行。while 循环通常用于下述情况，在循环开始之前不知道重复执行一个语句或语句块的次数。该语句语法格式如下所示：

```
while(表达式)
{
    //执行的语句块
}
```

下面创建一个示例，如代码 2.24 所示。

代码 2.24　使用 while 循环语句

```
<?php
    $i = 1;
    while ($i>0)
    {
        $i++;
        if ($i == 2)
        {
            echo $i;
            break;
        }
    }
?>
```

执行代码 2.24，结果如下所示：

```
2
```

3．do…while 语句

do…while 循环是 while 循环语句的扩展。do…while 语句与 while 语句唯一的区别在于，do…while 语句不管表达式的结果为真还是为假，循环语句至少执行一次。因此 do…while 循环适合于至少执行一次循环体的情况，其语法如下：

```
do
{
    //执行的语句块
}
while(表达式);
```

下面创建一个示例，具体说明 do…while 语句的使用，如代码 2.25 所示。

代码 2.25　使用 do…while 语句

```
<?php
```

```
    $i = 10;
    do
    {
      echo $i;
    } while ($i > 10);
?>
```

从代码 2.25 中可以看到，i 的初始值为 10，那么布尔返回值应该为 false，但还是会输出结果。也就是说，do…while 语句先不管表达式的结果是真是假，都会在执行一次后才再判断表达式的结果。该段代码执行结果如下所示：

```
10
```

4．foreach 循环语句

foreach 循环可以迭代出集合中的每一项，但并不能修改集合中的任何一项，是可以用于遍历数组的循环语句，从 PHP 5.0 开始，也可以用于对象。foreach 的语法如下：

```
foreach(集合 as 变量名)
{
执行语句;
}
```

foreach 循环语句每执行一次，循环变量就读取集合中的一个元素。下面创建一个示例说明 foreach 语句的使用，代码如下所示：

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as $value)
{
    $value = $value * 2;
    echo($value);
}
?>
```

其中，foreach 语句循环的参数是由两个元素组成的表达式，用关键字 as 分隔，关键字左边为要访问的元素集合的名称，右边为临时变量。该示例输出结果如下所示：

```
2468
```

2.6.3 break 和 continue 语句

很多时候，程序需要从一个语句块跳转到另一个语句块，因此 PHP 提供了两个可以立即跳转程序的语句，这两个跳转语句是 break 语句和 continue 语句。程序员使用跳转语句可以更加精确地控制整个流程，方便程序的设计，本节将详细介绍这两种跳转语句。

1. break 跳转语句

break 跳转语句表示程序的控制权强制性地从当前的语句块（主要为循环语句）中跳转出来，执行语句块下面的语句。break 可以结束 for 循环语句、foreach 循环语句、while 循环语句、do…while 循环语句和 switch 条件语句的执行。下面使用 break 跳转语句创建一个示例，如代码 2.26 所示。

代码 2.26　使用 break 跳转语句

```
<?php
$arr = array("唐晓阳","郑东方","裴亚敏");
foreach ($arr as $value)
 {
    echo($value);
    if($value=="郑东方")
    {
        break;
    }
}
?>
```

执行代码 2.26，结果如下所示：

```
唐晓阳郑东方
```

2. continue 跳转语句

continue 语句有些类似于 break 语句，但是只能出现在循环体中。该语句与 break 语句的区别在于：continue 并不是中断循环语句，而是中止当前迭代的循环，进入下一次的迭代。下面使用该跳转语句创建一个示例，如代码 2.27 所示。

代码 2.27　使用 continue 跳转语句

```
<?php
  for ($i = 0; $i < 5; ++$i) {
      if ($i == 2)
       {
         continue;
         print "$i\n";
         }
      echo($i);
  }
?>
```

执行代码 2.27，结果如下所示：

```
0134
```

2.7 函数

函数是一段完成指定任务的已命名的代码，可以根据给它的一组值或参数完成指定任务，并且可以返回一个值，也可以说是为了解决一些常见问题而制作好的“模板”。本节主要讲解如何定义函数和调用函数。

2.7.1 调用 PHP 函数

在 PHP 中，函数可以分为系统内部函数和自定义函数。不管函数是内置函数还是自定义函数，所有的函数都用相同的方法调用和求值，本节主要讲解如何在程序中调用函数。

在 PHP 中提供多种类型的内置函数，如文件函数，数据库函数，数学函数等。这样避免了处理问题编写重复的代码。函数的使用，达到了代码重用的目的。同样也可以自己编写函数，放入到函数库中作为自己的内置函数使用。本节主要讲解如何在 PHP 应用程序中调用这些内置函数，这些内置函数的意义将在下面的章节中详细讲解。下面创建一个调用 PHP 内置函数的示例，如代码 2.28 所示。

代码 2.28　调用内置函数示例

```
<?php
   echo "使用 rand 函数产生一个随机数";
   echo rand();
   echo "<br/>使用 round 函数求取一个整数";
   echo round(7.25);
   echo "<br/>使用 floor 函数求取一个整数";
   echo floor(7.25);
   echo "<br/>使用 ceil 函数求取一个整数";
   echo  ceil(7.25);
   echo "<br/>使用 max 函数求取两个数的最大值";
   echo  max(5,10);
   echo "<br/>使用 sqrt 函数求取一个数的平方根";
   echo sqrt(6);
?>
```

以上示例调用了 PHP 中的内置函数，调用自定义函数和调用内置函数相同，在此就不再详细讲解。保存并执行代码 2.28，结果如下所示：

```
使用 rand 函数产生一个随机数 19787
使用 round 函数求取一个整数 7
使用 floor 函数求取一个整数 7
使用 ceil 函数求取一个整数 8
使用 max 函数求取两个数的最大值 10
使用 sqrt 函数求取一个数的平方根 2.4494897427832
```

2.7.2　用户自定义函数

PHP 本身提供了大量的函数，使用这些函数可以完成任务中的大部分工作。但是，有一些个性化的需求 PHP 自带的函数无法完成，必须自己来写一些函数，处理这些个性化的需求。本节主要讲解如何在 PHP 中定义自定义函数。

在 PHP 中，定义一个自定义函数可以使用下面的语法格式：

```
function [&] function_name(参数)
{
    //执行程序代码
}
```

在定义自定义函数时，函数名可以是以字母或下划线开头后跟 0 个或者多个字母、下划线或数字的任何字符串。在 PHP 中函数名不区分大小写，可以以 fun(0)、Fun(0)和 FUn(0)调用 Fun(0)函数，因为这些函数指的都是同一个函数。函数通常会返回一些值，在自定义函数中，可以使用 return 返回相应的值。下面创建一个示例，具体讲解如何在 PHP 中定义自定义函数，如代码 2.29 所示。

代码 2.29　创建自定义函数

```
<?php
function Sum($i,$z)
{
    $s = $i + $z;
    return $s;
}
?>
```

在代码 2.29 中，函数有两个 int 类型参数，将这两个参数相加后返回其和，这就是该函数的功能。

以往，如果需要让字体显示为带下划线的粗斜体，至少要写这样一段代码<u><i> YOUND</i></u>。当然，如果网站中仅有一处需要让字体变粗变斜，大家就可用这段代码。但是如果在网站中有几十处要用到粗斜体，那么就需要书写几十遍这样的代码，显然，为了提高效率，要用一种相对简单的方法来实现这个需求，那就是自定义函数。下面创建一个示例，如代码 2.30 所示。

代码 2.30　使用自定义函数

```
<?php
function Get_Bold_Italics_underline($text)
{
 $text = "<u><i><b>$text</b></i></u>";
 return($text);
}
```

```
    echo Get_Bold_Italics_underline("欢迎光临!<a href='http://www.itzcn.com'>窗内
    网</a>");
    echo "<br/>";
    echo "欢迎光临!<a href='http://www.itzcn.com'>窗内网</a>";
    echo "<br/>";
    $MySpace="欢迎光临!<a href='http://www.itzcn.com'>窗内网</a>";
    echo Get_Bold_Italics_underline($MySpace);
    ?>
```

下面对自定义函数进行测试，在参数中直接输入要加下划线的粗斜体文字，或者是将文字放入一个变量中，然后再代入参数。执行程序页面效果如图 2-3 所示。

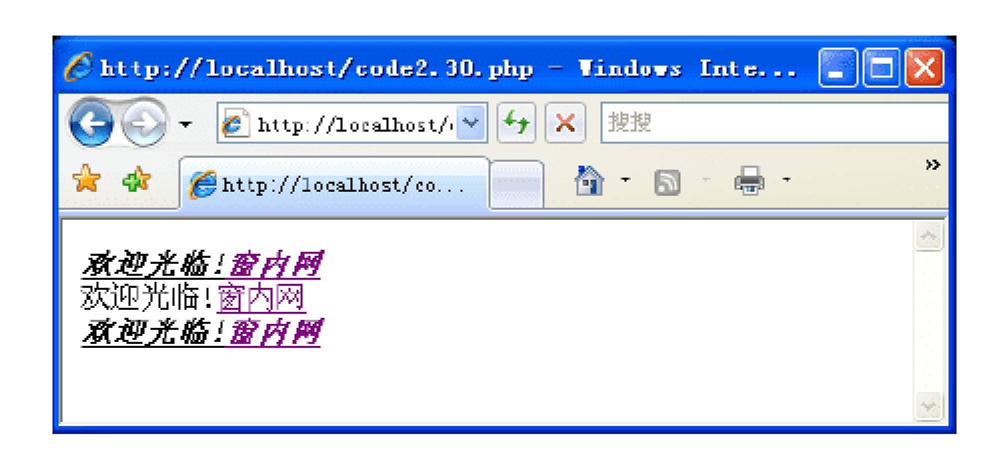


图 2-3　自定义函数

2.7.3　函数库

在 PHP 中有多种类型的内置函数，例如数学函数、数据库函数和文件函数等。本节主要介绍一些重要函数的意义和功能。

1. Math 数学函数

在 PHP 内置函数中，数学函数仅能处理在计算机上 integer 和 float 范围内的值。使用该类别函数无需安装、无需编译、无需配置，就可以直接使用，这是因为数学函数是作为 PHP 内核的一部分。如下所示为 PHP 中常见的内置数学函数。

- **abs()**　绝对值;
- **asin()**　反正弦;
- **ceil()**　进一法取整;
- **decbin()**　十进制转换为二进制;
- **floor()**　舍去法取整;
- **max()**　找出最大值;
- **min()**　找出最小值;
- **pow()**　指数表达式;
- **rand()**　产生一个随机整数;
- **round()**　对浮点数进行四舍五入;
- **sin()**　正弦;
- **sqrt()**　平方根。

2. 日期/时间函数

在 PHP 中，存在一类日期时间函数，可以用这些函数得到 PHP 所运行的服务器的日期和时间，并可以用这些函数将日期和时间以很多不同方式格式化输出。本节将主要介绍这些日期和时间函数，如下所示。

- **checkdate($month,$date,$year)**　该函数主要用于检测日期是否合法，经常在计算或保存日期到数据库之前使用。
- **getdate($ts)**　在没有自变量的情况下，该函数以结合数组的方式返回当前日期与时间。数组中的每个元素代表日期/时间值中的一个特定组成部分。可向函数提交可选的时间标签自变量，以获得与时间标签对应的日期/时间值。
- **date($format, $ts)**　可用在一系列的修正值中，将整数时间标签转变为所需的字符串格式。
- **strtotime($str)**　此函数将可阅读的英文日期/时间字符串转换成 UNIX 时间标签。应用此函数将非标准化的日期/时间字符串转换成标准、兼容的 UNIX 时间标签。
- **microtime()**　将 UNIX 时间标签格式化成适用于当前环境的日期字符串。
- **gmdate($format, $ts)**　此函数将 UNIX 时间标签格式化成日期字符串。
- **date_default_timezone_get()或 date_default_timezone_set($tz)**　此函数返回脚本中所有日期时间函数所使用的默认时区。

2.8　数组

数组为一组具有共同特性的元素，每个元素都由一个特殊标识符区分。在 PHP 程序中使用数组可以存储、操作、排序和获取数据集，并且还可以使用数组处理函数检查给定数组中是否存在特定对象、对数组元素计数、增加或删除元素，或对元素排序等。本节主要讲解数组概念和处理数组的一些函数。

2.8.1　数组概述

在 PHP 中，数组结构中可以包括完全无关的元素，甚至可以不属于同一种类型。例如，一个数组可能包括学生的姓名、性别、联系电话、考试成绩或驾驶证编号等一些不相关的元素。下面定义了一个数组，如下所示：

```
$East[0]="唐晓阳";
$East[1]="0461411";
$East[2]=88;
$East[3]=66;
$East[4]="河南、郑州";
$East[5]="窗内网: http://www.itzcn.cn";
```

上述数组$East 中索引号为 0、1、4、5 的元素所存储的是姓名、学号、籍贯以及网址，

其类型都为字符串；索引号为 2、3 的元素所存储的是数学成绩及计算机成绩，但是其类型却是整数；在这个数组中索引号为 5 的元素值与前面其他元素值没有任何关系，但它同样存储在了该数组中。在 PHP 中，通常使用数组元素存储不同类型的数据，这为处理数据及定义数据类型提供了方便。

2.8.2 管理数组

管理数组主要是对数组元素进行增加及删除、定位数组元素以及确定数组大小及其唯一性。这些操作涉及了相当多的数组函数，本节将详细介绍这些管理数组的函数。

1. 增加和删除数组元素

在实际的应用中，数组元素的增加和删除是最常见的操作。在 PHP 中，使用其处理数组的内置函数可以实现此功能，下面详细介绍对数组元素进行增加和删除的函数。

- **array_push()函数** 该函数将 variable 增加至 target_array 的末尾，成功时返回 true，否则返回 false。为此函数传递多个变量作为输入参数，将同时向数组压入多个变量（元素）。该函数语法格式如下所示：

```
int array_push(array target_array,mixed variable[,mixed variable…])
```

下面使用该函数创建一个示例，代码如下所示：

```
$MyFriends = array("唐晓阳","裴亚敏");
array_push($MyFriends,"东方","海洋");
```

等价于：

```
$MyFriends=array)("唐晓阳","裴亚敏","东方","海洋");
```

- **array_pop()函数** 该函数用于返回 target_array 的最后一个元素，并在结束后重置数组的指针。该函数的使用格式如下所示：

```
mixed array_Top(array target_array)
```

使用 array_pop()函数的示例如下所示：

```
$MyFriends = array("唐晓阳","王洋","裴亚敏","郭海霞","凌云");
$oneFriend = array_pop($MyFriends);
```

等价于：

```
$oneFriend = "凌云";
```

- **array_shift()函数** 该函数类似于 array_pop()，只是它返回 target_array 的第一个数组元素，而非最后一个。其结果是，如果使用的是数值键，则所有相应的值都会下移，而使用关联键的数组不受影响。与 array_pop()一样，array_shift()也会在结束时重置指针。该函数的使用格式如下所示：

```
mixed array_shift(array target_array)
```

使用 array_shift()函数的示例如下所示：

```
$MyFriends = array("唐晓阳","裴亚敏","郑圆圆","唐军","魏海洋");
$oneFriend = array_shift($MyFriends);
```

等价于：

```
$oneFriend = "唐晓阳";
```

- **array_unshift()函数** 该函数与 array_push()函数相似，只是它将元素增加到数组头，而不是尾。所有已有的数值键都会相应地修改，反映出在数组中的新位置，而关联键不受影响。该函数的语法格式如下所示：

```
int array_unshift(array target_array,mixed variable[,mixed variable…])
```

使用 array_unshift()函数的示例如下所示：

```
$MyFriends = array("唐晓阳","裴亚敏","郑圆圆","唐军","魏海洋");
array_unshift($MyFriends,"李振国","郝军");
```

等价于：

```
$MyFriends = array("李振国","郝军","唐晓阳","裴亚敏","郑圆圆","唐军","魏海洋");
```

- **array_pad()函数** 该函数会修改 target 数组，将其大小增加到 length 指定的长度。这通过在数组中填充由 pad_value 指定的值实现。如果 pad_value 是正数，将填充到数组的右侧（后面）；如果为负，则将填充到左侧（前面）。如果 length 等于或小于当前大小，将不做任何操作。该函数的语法格式如下所示：

```
array array_pad(array target,integer length,mixed pad_value)
```

使用 array_pad()函数的示例如下所示：

```
$MyFriends = array("唐晓阳","裴亚敏","郑圆圆","唐军","魏海洋");
$MyFriends = array($MyFriends,7, "郝军");
```

等价于：

```
$MyFriends = array("唐晓阳","裴亚敏","郑圆圆","唐军","魏海洋","郝军","郝军");
```

2. 定位数组元素

定位数组元素指的是用户通过一些函数有效筛选数组中的数据，下面详细介绍一些常用的定位数组元素函数。

- **in_array()函数** 该函数用于检查数组中是否存在某个值。它在 haystack 中搜索 needle，如果找到则返回 true，否则返回 false。如果第三个参数 strict 的值为 true，则 in_array()函数还会检查 needle 的类型是否和 haystack 中的相同。该函数的使用格式如下所示：

```
bool in_array(mixed needle, array haystack [, bool strict])
```

使用 in_array()函数的示例如下所示：

```
$ages = array(18,19,20,21,22,23,24,25,26,27,28,29,30);
if (in_array("24",$ages))
{echo "在数组中存在 24 <br/>";}
if (in_array(24,$ages,1))
{echo "类型相同";}
```

上述代码成功执行后，将输出如下信息：

```
在数组中存在 24
类型相同
```

从执行结果可以看出，第二个测试要求数据类型必须匹配。由于第二个测试完成的是整数和整数的比较，所以测试成功，如果它们类型不同将导致失败。

- **array_keys()函数** 该函数用于返回数组中所有的键名，也就是返回一个由数组 target_array 中所有键组成的数组。如果指定了可选参数 search_value，则只返回该值的键名。该函数的使用格式如下所示：

```
array array_keys ( array target_array [, mixed search_value] )
```

使用 array_keys()函数的示例如下所示：

```
<pre>
<?php
$array = array (0 => 25, "name" => "yound");
print_r(array_keys ($array));
$array = array ("tang", "yound", "ya", "tang", "tang");
print_r(array_keys ($array, "tang"));
?>
</pre>
```

上述代码成功执行后，将输出如下信息：

```
Array
(
    [0] => 0
    [1] => name
)
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
```

- **array_keys_exists()函数** 该函数用于检查给定的键名或索引是否存在于数组中，它在给定的 key 存在于数组中时返回 true。key 可以是任何能作为数组索引的值。该函数的使用格式如下所示：

```
bool array_key_exists ( mixed key, array target_array)
```

使用 array_keys_exists()函数的具体示例如下所示：

```
$search_array = array("name" => "yound", "sex" => "男");
if (array_key_exists("name ", $search_array)) {
   echo "name 键在该数组中";
}
```

执行上述代码，结果如下所示：

```
name 键在该数组中
```

- **array_values()函数**　该函数用于返回 target_array 数组中所有的值并给其建立数字索引。该函数的使用格式如下所示：

```
array array_values(array target_array)
```

使用 array_values()函数的具体示例如下所示：

```
$array = array ("name" => "yound", "sex" => "男");
print_r(array_values ($array));
```

执行上述代码，结果如下所示：

```
(
    [0] => yound
    [1] => 男
)
```

- **array_search()函数**　该函数用于在数组中搜索给定的值，如果成功则返回相应的键名。该函数的使用格式如下所示：

```
mixed array_search ( mixed needle, array haystack [, bool strict] )
```

该函数在 haystack 中搜索 needle 参数并在找到的情况下返回键名，否则返回 false。同前面一样，如果可选的第三个参数 strict 为 true，则 array_search()还将在 haystack 中检查 needle 的类型。

3．确定数组大小和唯一性

在 PHP 中，有些函数可以用来确定数组中元素总数以及唯一值的个数。接下来将对这些函数进行介绍，如下所示。

- **count()函数**　该函数用于返回 input_array 中元素的总数。如果启用了可选的 mode 参数，数组将进行递归计数。在统计多维数组中所有元素的个数时这个特性很有用。该函数的使用格式如下所示：

```
int count (array input_array [, int mode] )
```

使用 count()函数的具体示例如下所示：

```
$MyFriends = array("唐晓阳","裴亚敏","郑圆圆","唐晓亮","杨阳");
echo count($MyFriends);
```

执行上述代码，结果如下所示：

```
5
```

count()函数的另一种示例如下所示：

```
$StuName = array ("唐晓阳","裴亚敏",array ("郑圆圆","唐晓亮","杨阳"));
echo count($StuName,1);
```

上述语句代码统计了$StuName 中的标量元素个数和数组个数。保存该段代码，执行结果如下所示：

```
6
```

- **array_count_values()函数** 该函数用于统计数组中所有的值出现的次数。该函数返回一个包含关联键/值对的数组。返回数组的每个键表示 input_array 中的一个值，相应的值表示这个键在 input_array 中出现的频度。该函数的语法格式如下所示：

```
array array_count_values(array input_array)
```

使用 array_count_values()函数的具体示例如下所示：

```
$array = array(5, "East", 5, "East", "Joy",5);
print_r(array_count_values ($array));
```

执行上述代码，结果如下所示：

```
Array
(
    [5] => 3
    [East] => 2
    [Joy] => 1
)
```

- **array_unique()函数** 该函数会删除 input_array 中所有重复的值，返回一个由唯一值组成的数组。array_unique()函数先将值作为字符串排序，然后对每个值只保留第一个遇到的键名。这并不意味着在未排序的 array 中同一个值的第一个出现的键名会被保留。该函数的语法格式如下所示：

```
array array_unique ( array input_array)
```

使用 array_unique()函数的具体示例如下所示：

```
$input = array("East" => "东方", "tang", "Joy" => "晓阳", "tang", "Yound");
$result = array_unique($input);
print_r($result);
```

执行上述代码，结果如下所示：

```
Array
(
    [East] => 东方
    [0] => tang
    [Joy] => 晓阳
    [2] => Yound
)
```

第3章 面向对象的 PHP



面向对象编程（OOP）是编程的一项基本技能，如何使用 OOP 的思想来进行 PHP 的高级编程，对于提高 PHP 编程能力和规划好 Web 开发构架都非常有意义。PHP 5.0 完全重新开发了处理对象部分的内核，提供了更多功能的同时也提高了性能，例如 PHP 中对接口的处理方式。面向对象程序设计以“数据控制访问代码”为主要原则，围绕数据来组织程序。在进行面向对象编程时需要定义数据和作用于数据上的方法。本章将详细介绍面向对象的基础，例如类、对象和方法等。

学习目标 Objective

- 建立面向对象思想
- 熟悉 PHP 面向对象的特点
- 灵活掌握类和对象
- 熟练创建和使用字段
- 掌握属性和常量
- 掌握如何创建和使用方法
- 了解构造函数和析构函数
- 掌握 PHP 静态成员的创建和使用

3.1 OOP 特性

面向对象程序开发是一种软件开发思想，它将数据和对数据的操作作为一个相互依赖不分割的整体，也可以说将对数据的操作作为一个对象进行处理，采用数据抽象和信息隐蔽技术，以使软件开发简单化，符合人们的思维习惯，有助于控制软件的复杂性，提高软件的生产效率，从而得到了广泛的应用，已经成为目前最为流行的一种软件开发方法。面向对象的编程语言更加接近物质世界。OOP 的内容主要涉及到 3 大基本概念：封装、继承和多态。

3.1.1 封装

在现实世界中，可以将身边的一切作为对象来看待。如办公室的计算机、书柜、马路奔

跑的汽车以及路上的行人。这些存在的实体，都在执行自己的动作，或者以某种形态存在。整个世界由无数个独立的实体所组成，不需要了解每个实体的创建过程，运行方式。例如空调是一个实体，不需要知道空调是用什么材料制作而成，它的制作流程是什么，只需要知道可以通过遥控器控制空调的温度。面向对象程序设计的思维模式就是调用另外一个存在的实体时，不需要知道该实体的实现细节，只需要知道使用该实体的通用接口。在 OOP 的程序中，存在无数个类似现实世界中的实体，这些实体在应用程序中称为类。在每一个实体中，都会存在相应的行为和特点，其他的实体可以通过相应的接口来操作这个实体。

一般来说封装就是隐藏类的数据成员，只向外提供一些公用的操作接口，只能通过这些接口来操作类的数据成员，而不能直接对这些数据成员进行赋值改变等操作。这样做的好处在于，如果选择直接暴露数据成员，有些人可能会将这些成员修改为非法数据导致程序出错。所以封装了操作数据成员的细节，能够对新值进行验证或者进行其他操作等。封装是将代码及其处理的数据绑定在一起的编程机制，该机制保证了程序和数据都不受外部干扰且不被误用。

3.1.2　继承

在现实世界中，公共汽车是汽车的一种，汽车又是车的一种，车又是机械的一种。如果不使用层次的概念，每个对象都需要明确定义各自的全部特征。如果通过层次分类方式，一个对象只需要在其类中定义它所特有的属性和行为，然后从父类中继承通用属性。正是由于继承机制，才使得一个对象可以成为一个通用类的特定实例，一个深度继承的子类将继承它在类层次中的每个祖先的所有公有属性。

继承是面向对象程序设计的主要特征之一，它可以让开发人员重用代码，可以节省程序设计的时间。继承就是在类之间建立一种相交关系，使得新定义的派生类的实例可以继承已有基类的特征和能力，而且可以加入新的特性或者是修改已有特性，然后建立起类的新层次。继承也可以指一个对象从另一个对象中获得属性的过程，它支持按层次分类的概念。目前 PHP 不支持多重继承，只支持单继承，所以不能从两个或两个以上类中派生新的类。面向对象的开发方法建立在继承概念的基础上。这种策略提高了代码的可重用性，因为它使得人们能够在多个应用程序中使用设计良好的类。继承的使用，也提高了类之间的层次性。

继承与封装可以互相作用，提高了软件模块的可复用性和可扩充性，还可以提高软件的开发效率，也能够利用前人或自己以前的开发成果，同时在自己的开发过程中还能够有足够的灵活性，不拘泥于复用的模块。例如，一个给定的类封装了某些属性，其任何子类将会含有同样的属性，再加各个子类所特有的属性。这是面向对象程序在复杂性上呈线性而非几何增长的一个重要概念，新的子类继承其所有祖先的所有属性，子类和系统中的其他代码不会产生无法预料的交互作用。

3.1.3　多态

多态是指同一操作作用于不同的对象，可以有不同的解释，产生不同的执行结果。其实，它的真正意义在于在实际开发中，只需要关注一个接口或基类的编程，而不必担心一个对象

所属的具体类。例如，有一个停车场，里面停放了 10 辆轿车，但是刚刚又停放了 5 辆越野车和一辆大客车，一共有多少汽车？此时，需要将所有的汽车加起来，结果是共有 16 辆汽车。为了计算将 3 种不同种类的汽车当成一种通用类型（汽车）对待，在这里即使用了多态性。映射到程序中，可以使用一个对象来完成不同环境下的功能操作。

多态是指 OOP 能够根据使用类的上下文来重新定义或改变类的性质或行为。多态是对象的一种能力，它可以在运行时刻根据传递的对象参数决定调用哪一个对象的方法。例如，考虑一个叫 Person 的类，可以用名称为 David，Charles 和 Alejandro 的类来子类化 Person。Person 有一个抽象方法 AcceptFeedback()，所有的子类都要实现这个方法。这意味着，任何使用基类 Person 的子类的代码都能调用方法 AcceptFeedback()，不必检查该对象是一个 David 还是一个 Alejandro，仅知道它是一个 Person 就够了。

在这个示例中的 Person 类也可以被创建为一个接口。当然，与上面相比存在一些区别，主要在于：一个接口并没有给出任何行为，而仅确定了一组规则。一个 Person 接口要求的是“必须支持 AcceptFeedback()方法”，而一个 Person 类可以提供一些 AcceptFeedback()方法的省略代码，如果能够简单地勾勒出类所要实现的一组期望的功能，那么也可以使用一个接口。

3.2　关键的 OOP 概念

本节主要介绍面向对象的基础性概念，例如如何创建类、如何实例化对象、类中的字段和方法等。

3.2.1　类和对象

类实际上就是一个模板，它定义了某个概念或真实事物的性质和行为。这个模板提供了一个基础，可以在此基础上创建实体的特定实例，这些特定的实例称为对象。对象是类的概念实现，或者说是类实例化后的结果。如一辆轿车就是汽车类的一个实例化对象，轿车具有汽车的行为和特点。

同样，在 PHP 程序中，也可以创建一个类，这时类只是具有一些概念的模型，即类中定义了相应的方法和字段，而没有实现这些概念。在 PHP 中创建一个类的语法格式如代码 3.1 所示。

代码 3.1　类的创建格式

```
class Person
{
  Access Variable Declaration
  Access Function Declaration
}
```

在上述代码中，类名是 Person，类中的语句包含在一对大括号中，一个类由成员方法和成员字段组成。类的创建示例如代码 3.2 所示。

代码 3.2　创建好的 Person 类

```
<?php
class Person
   {
        public  $name;
        public  $age;
        function AddPerson($personName,$personage)
        {
         $this->name = $personName;
         $this->age = $personage;
         printf("你的姓名是: %1\$s<br/>你的年龄是: %2\$s<br/>",$this->name,
         $this->age);
         if($this->age>21)
         {
             echo ("你可以加入我们团队");
         }
         else
         {
             echo ("你还小，不能参加我们的团队工作");
         }
        }
   }
   $p = new Person();
   $p->AddPerson("唐晓阳",18);
?>
```

在代码 3.2 中，定义了一个 Person 类，该类为 Person 设置两个字段，分别表示 Person 的姓名和年龄，还有一个加入团队的函数。在 PHP 中，可以使用 new 关键字创建一个对象，如代码 3.2 中加粗部分。对于该示例所使用的语句，将在后面的章节中讲解。当对象创建完成之后，就可以直接调用类中方法和字段，一个类可以创建多个对象。将该段代码保存到 DemoClass.php 文件中，并把该文件保存到 F:\MyWeb\Apache\htdocs 目录下，打开 IE 浏览器，在地址栏中输入 http://localhost/DemoClass.php 并单击【转到】按钮，页面效果如图 3-1 所示。

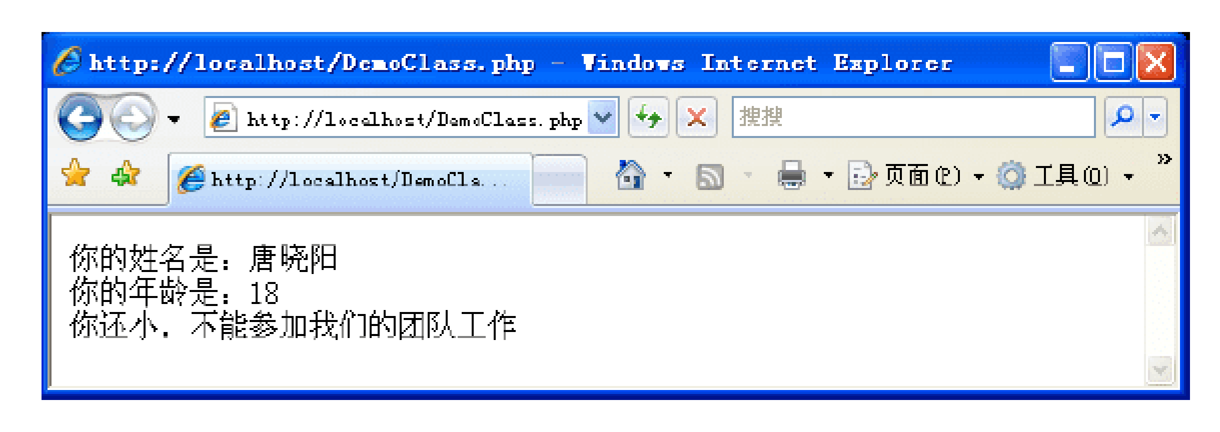


图 3-1　页面效果图

3.2.2　字段

字段是用来描述类各个方面的性质。它与一般的变量非常相似，只是有一些细微的差别。

本节主要介绍如何声明字段、使用字段和类中字段的作用范围。

1. 声明字段

类中的字段，实际上就是对象中所具有的属性，用来表示实体的某一种状态。在 PHP 中创建一个字段和创建一个变量基本上相同，只不过创建的位置不同。PHP 是一种弱类型的语言，字段可以不需要声明，直接创建并赋值，但是很少这样使用。也可以在类中先声明，再赋值。通常都是在类的开始声明字段。下面创建一个类中声明字段的示例，如代码 3.3 所示。

代码 3.3　类中声明字段

```
class Employee
{
    public $EmployeeName = "Yound Tang";
    private $EmployeeAge;
    public $EmployeeSex;
}
```

在代码 3.3 所示代码中，创建了 3 个字段$EmployeeName、$EmployeeAge 和$EmployeeSex。public 和 private 表示这个字段的作用域范围，声明字段之后可以在作用域描述符范围内使用。创建的第一个变量直接为其赋值，而后两个字段都没有为其赋初值。

2. 使用字段

与变量不同的是，不能直接使用类中的字段，需要使用“->”操作符调用字段，并且只能使用对象调用。所以在使用字段时，还需要指定相应的对象，使用字段的语法格式如下所示：

```
Object->field
```

在定义字段的类中引用字段时，同样也需要使用“->”操作符，但此时不使用相应的类名，而是使用 $this 关键字，该关键字表示要引用当前类中的字段。下面创建类中和类外引用字段的示例，如代码 3.4 所示。

代码 3.4　引用字段

```
<?php
  class Automotive
  {
     public $speed;
     public $type;
     function Driving()
     {
         printf("%1\$s 的速度是%2\$s",$this->type,$this->speed);
     }
  }
  $a  = new Automotive();
```

```
    $a->speed = "100 千米/小时";
    $a->type = "公共汽车";
    $a->Driving();
?>
```


在代码 3.4 中，创建了一个类 Automotive，并在类中创建了两个字段，分别为$speed 和$type，下面又为该类实例化了一个对象，利用该对象并使用操作符“->”为对象中字段赋值，最后调用 Driving()方法。执行这段代码会得到如下所示结果：

```
公共汽车的速度是 100 千米/小时
```

3．字段的作用域

在 PHP 中，字段共有 5 个相应的作用域范围，用来表示类的字段可以影响的空间，其值分别为：public、private、protected、final 和 static，static 作用域将在 3.4 节中详细介绍。其详细介绍如表 3-1 所示。

表 3-1　访问控制修饰符

关键字	说明
public	表示修饰的字段的作用域范围是公共作用域
private	表示修饰的字段只能在类中访问，类以外的其他位置无法访问该字段
protected	表示修饰的字段是受保护的。受保护的字段只能在类中调用，不允许在类的外部调用这些字段
final	当设置为 final 时，将阻止在子类中覆盖这个字段
static	静态作用域

- **public**　该作用域的字段可以在类外部直接通过对象名访问，并进行修改。下面创建一个示例，具体介绍 public 关键字的应用，如代码 3.5 所示。

代码 3.5　使用 public 关键字

```
<?php
 class Student
 {
    public $stuName;
    public $stuAge;
 }
 $stu = new Student();
 $stu->stuName = "唐晓阳";
 $stu->stuAge = 23;
 printf("你的姓名是：%1\$s<br/>你的年龄是：%2\$s",$stu->stuName ,$stu->stuAge);
?>
```

在代码 3.5 中，首先创建了 Student 类，并在类中创建了两个具有公共作用域的变量$stuName 和$stuAge，并且没有赋初值。然后在类的外部，创建类 Student 的对象$stu，并给字段 stuName 和 stuAge 赋值，最后输出这两个字段。该示例执行结果如下所示：

```
你的姓名是：唐晓阳
你的年龄是：23
```


在类中，声明某个字段是公共作用域，方便了对该字段的使用和修改。但是在 OOP 中并不鼓励使用公共字段，因为这样做，会将程序的一些细节暴露在类的外部，并且直接访问类的某些数据，会忽略对某些数据的数据验证。例如无法阻止用户修改字段的值。对于这样的情况，通常采用两种方法解决，一种办法是将数据封装在对象中，只通过一些称为公共方法的接口来访问，这种封装的数据具有私有作用域。第二种方法是使用属性。

- **private** 又称私有的访问控制修饰符，是限制最为严密的控制关键字。其示例如代码 3.6 所示。

代码 3.6 声明私有字段

```
<?php
class Exam{
   private $name;
   private $telephone;
}
?>
```

在上述代码中，创建了两个私有字段 name 和 telephone，该字段不能通过该类的实例化对象来访问，也不能通过该类的子类来访问。如果需要在类的外部使用这些字段的值，可以通过保护作用域来完成。私有字段必须通过公共接口来访问，这符合前面介绍的封装概念。私有字段的使用如代码 3.7 所示。

代码 3.7 使用私有字段

```
<?php
   class staff{
        private $name;
        public function setName($name){
            $this->name=$name;
         }
    }
   $sta=new staff();
   $sta->setName("Hello");
?>
```

在上述示例中，创建了类 staff，并在类中创建了一个私有字段 name，创建了一个公共方法 setName()，用来设置私有字段 name 的值。在下面创建该类的对象$sta，并调用该类的公共接口设置私有字段的值。

- **protected** 表示修饰的字段是受保护的。受保护的字段只能在类中调用，不允许在类的外部调用这些字段。其使用示例如下所示：

```
class  Example{
    protected  $string;
```

```
    }
```

和私有字段的区别在于，在继承的子类中，可以访问这些受保护的字段，这是私有字段不能够具备的。如果在子类中试图访问父类中的私有字段，将会导致致命的错误。因此，如果希望扩展该类，就应当使用保护字段而不是私有字段。

- **final**　把字段设置为 final 作用域时，将阻止在子类中覆盖这个字段，如下所示为声明字段为 final 的示例。

```
class Student
 {
    final $stuName;
 }
```

3.2.3　属性

OOP 提供了强大的功能，属性就是一个例子，属性是受保护的字段。通过强制在方法中访问和操作字段，一方面保护字段，同时另一方面允许像访问字段一样访问数据。这些方法称为访问方法和修改方法，或非正式地称为获取方法（getxxx()）和设置方法（setxxx()）。它们将会分别在访问或操作字段时自动触发。

在 PHP 5.0 中，没有提供和其他 OOP 语言一样处理并使用属性的功能。如果要实现该功能，不能够直接实现，需要使用公共方法模拟这种功能。需要声明两个函数 getName()和 setName()，为属性 name 分别创建获取方法和设置方法，并可以在函数中设置相应的语句实现功能。

在 PHP 5.0 中，通过重载__set()方法和__get()方法也能实现对属性的处理。在 PHP 的类的执行过程中，如果在代码中试图引用一个类定义不存在的成员变量，就会自动触发这些方法，并执行方法里面的语句。属性有很多用途，如产生错误消息，甚至通过动态创建新的变量来扩展类。下面详细讲解__set()和__get()，还有创建自定义获取方法和设置方法。

1. __set()

该方法主要负责隐藏字段的赋值实现，并在为类字段赋值之前验证类数据。它接受一个属性名和相应的值作为输入，如果方法成功执行就返回 true，否则返回 false。可以称为修改方法或者设置方法。该方法的语法格式如下：

```
boolean __set([String property_name,][mixed value_to_assign])
```

在这里需要注意的是，set 函数前面的下划线是两个而不是一个，如果格式不正确，该方法是不会执行的。现在创建一个示例，演示一下该方法的使用，如代码 3.8 所示。

代码 3.8　使用__set()方法

```
<?php
  class staff{
        var $name;
```

```
        function __set($propName,$propValue){
          echo '该变量不存在';
          echo "<br/>";
         }
    }
  $em=new staff();
  $em->name="唐晓阳";
  $em->title='hello';//该属性不存在
  echo $em->name;
?>
```

将上述代码保存，名称为 Demo-shuxing.php，并保存在指定的位置。打开 IE 浏览器，在地址栏中输入 http://localhost/Demo-shuxing.php，单击【转到】按钮，显示结果如下所示：

```
该变量不存在
唐晓阳
```

在本例中，创建了一个类 staff，在类中创建一个属性$name，并重写了方法__set()用来检验要进行赋值的变量是否存在，如果该变量值不存在，就会执行__set()方法里面的语句。在下面创建了该类的一个对象，给 name 赋值，并给不存在的属性 title 赋值，当程序执行到此处时，就会触发执行__set()方法。

2. __get()

该方法主要负责封装获得类变量所需的代码。它接受一个属性名作为输入参数，即获取属性的值。它在成功执行时返回 true，否则返回 false。可以称作获取方法。该方法的语法格式如下所示：

```
boolean __get([string property_name])
```

该方法的使用示例如代码 3.9 所示。

代码 3.9　使用__get()方法

```
<?php
class foo
{
    function __get($name)
    {
        echo ("需要获取属性$name");
    }
}
$f = new foo();
$f->bar =10;
print ($f->Name);
echo ("<br/>");
print ($f->bar);
```

```
?>
```

在上述代码中，创建了一个类 foo，并在类中实现了__get()方法，如果该方法被触发，就执行语句。当试图输出一个没有经过定义的属性值时，就会触发__get()方法。执行该段程序，结果如下所示：


```
需要获取属性 Name
10
```

3. 创建自定义获取方法和设置方法

使用__set()和__get()方法可以对属性进行支持，但对于管理复杂面向对象应用程序中的属性而言，其功能还是不完善。可以采用另外一种方式来设置和获取字段的值。当在一个类中创建一个变量时，可以为该变量创建两个方法实现变量值的设置和获取。其使用示例如代码 3.10 所示。

代码 3.10　创建自定义获取方法和设置方法

```
<?php
class Student
{
    private  $stuClass;
    public function GetStuClass()
    {
        return $this->stuClass;
    }
    public function SetStuClass($stuClass)
    {
        $this->stuClass = $stuClass;
    }
}
$s = new Student();
$s->SetStuClass("计算机科学与技术系—08 届--一班");
echo ($s->GetStuClass());
?>
```

在代码 3.10 中，创建了一个 Student 类，并在该类中创建一个变量 stuClass，并创建了自定义设置和获取方法，即 GetStuClass()函数和 SetStuClass()函数。最后实例化了一个类的对象，使用该对象调用创建好的设置和获取方法。这种方式不如使用属性那么方便，但通过使用标准的命名约定封装了管理和存取任务，同时还可以向设置方法中加入一些校验的代码。

3.2.4　常量

常量就是在程序执行过程中不会改变的数值。在 PHP 中可以使用 const 关键字定义常量。对于类中的任何一个实例化对象，其运行过程中常量会一直存在，并且常量值在这些对象的

整个生命周期中都会保持不变。

常量不同于普通变量，常量值不能通过对象的实例来访问，而应使用$object::constant 表达式访问。常量值必须是一个常量表达式，而不是一个变量、一个类的成员、一个数学表达式或者函数调用的结果。下面创建一个声明常量并调用的示例，如代码 3.11 所示。

代码 3.11　创建常量并调用常量

```
<?php
class Area
{
    const num=3.1415;
    public function FirstSum($r)
    {
        return self::num * $r * $r;
    }
}
$s = new Area();
echo "Π的值为: ".Area::num."<br/>";
echo "所求圆面积是:  ";
echo $s->FirstSum(10);
?>
```

代码 3.11 在类 Sum 中创建了一个常量和一个求圆面积的方法。当在方法中调用这个常量时，不能使用$this 关键字，而应使用 self 关键字。在类外调用常量时直接使用类名调用。

3.2.5　方法

类的方法和 PHP 脚本程序中的函数比较相似，只不过方法用来定义类的行为。类中的方法可以完成指定的功能，并且可以有一个返回值；同样也可以接受输入参数，并对该数值做出校验，返回结果。在 PHP 页面中，调用一个函数直接可以通过函数名来实现，调用类的方法和调用函数一样，只不过前面需要加上对象名，调用方法的格式如下所示：

```
Object->method_name();
```

1. 声明方法

可以在类中创建一个方法，其语法格式和函数的创建相同，只不过类中的方法都必须使用 public、protected 或 private 等作用域进行定义。如果没有设置这些关键字，默认作用域为 public，声明方法的语法格式如下所示：

```
scope function methodName(){
  方法体
}
```

在上述代码中，scope 表示方法的作用域范围，methodName 表示方法的名称，大括号中

表示该方法的执行语句。下面创建一个声明方法的示例，如代码 3.12 所示。

代码 3.12　声明方法

```
class Comparison
{
    private $num1 = 50;
     private $num2 = 60;
     public  function Compare()
     {
         if($this->num1 > $this->num2)
         {
             return true;
         }
         else
         {
             return false;
         }
     }
}
```

在代码 3.12 中，创建了一个 Comparison 类，在该类中定义了两个字段，并声明了一个方法，该方法比较这两个数的大小，如果 num1 值大就返回 true，否则返回 false。由此可知，类中的方法可以操作类本身的字段。但是，方法并不只限于操作类的字段，还可以像函数一样传递参数。



对于公共方法，可以不显式地声明作用域，而只是像声明函数（没有任何作用域）那样声明方法。

2．调用方法

下面使用代码 3.12 中的类来讲解如何调用方法，调用方法的代码如代码 3.13 所示。

代码 3.13　使用类中的方法

```
$c = new Comparison();
if($c->Compare())
{
    echo "数值 num1 比 num2 大";
}
else
{
    echo "数值 num1 不比 num2 大";
}
```

由代码 3.13 知，要调用类中的方法首先要实例化类，然后通过对象调用类中的方法。

3. 方法作用域

PHP 中方法的作用域关键字有 6 种：public、private、protected、abstract、final 和 static，本节主要介绍前 5 种作用域关键字，static 关键字将在后面的章节中详细讲解，表 3-2 为方法关键字的扼要介绍。

表 3-2 方法关键字

关键字	说明
public	表示该方法是一个公共方法，在 PHP 页面的任何一个位置都可以通过对象名来访问该方法
private	表示该方法是私有的，只能在类的内部使用，不能通过类的实例化对象调用，也不能通过类的子类调用。如果某些方法是作为另外一些方法的辅助，可以声明该方法为私有的
protected	表示方法是一个受保护的方法，只能在类的本身和类的子类中使用。这类方法可以用来帮助类或子类完成内部计算
abstract	表示方法是一个抽象方法。抽象方法表示只具有方法名，而没有方法体的方法
final	表示最终的，不可改变的含义

❑ public

当使用 public 关键字创建类的方法时，表示该方法为公共方法。公共方法可以在任何位置任何时间访问，如果方法前面无任何关键字，该方法也表示公共方法。下面创建一个示例，演示如何声明和调用公共方法，如代码 3.14 所示。

代码 3.14 公共方法的声明和调用

```
<?php
  class Visitors{
     public function SayHello()
     {
          echo "你好~! <br/>";
     }
     function sayStudy()
     {
         echo "欢迎学习 PHP<br/>";
     }
   }
   Visitors::SayHello();
   $s = new Visitors();
   $s->SayHello();
   $s->sayStudy();
?>
```

将代码 3.14 保存为 Demo-PublicMethod.php。执行程序，页面显示结果如下所示：

```
你好~!
你好~!
```

```
欢迎学习 PHP
```

在代码 3.14 中，创建了一个类 Visitors，并在类中创建了两个方法，分别输出不同的字符串，这两个方法的作用域都是 public。调用这两个方法采用两种不同形式，一种是通过类名直接调用，例如语句“Visitors::SayHello();”，其调用操作符为::，另外一种形式是通过对象名调用，例如语句“$s->SayHello();”。

- **private**

如果要创建一个私有方法，只需要在方法的前面加上 private，在表 3-2 中对该关键字的特点已经进行了详细介绍，在此就不再赘述。下面创建一个声明私有方法的示例，如代码 3.15 所示。

代码 3.15　声明私有方法

```
<?php
class GetSum
{
    private function ComNum($num)
    {
        if($num>10)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public function SumorProduct($number)
    {
        if($this->ComNum($number))
        {
            $sum = $number+$number;
            echo "这个数字的 2 倍为: ".$sum;
        }
        else
        {
            $product = $number*$number;
            echo "这个数字的平方为: ".$product;
        }
    }
}
$g = new GetSum();
$g->SumorProduct(8);
?>
```

在代码 3.15 中，创建一个 GetSum 类，并创建了两个方法 ComNum()方法和 SumorProduct()方法。方法 ComNum()是一个私有方法，只能由类本身调用。ComNum()充当一个校验功能，对参数进行判断。在方法 SumorProduct()中，调用了 ComNum()进行判断。这里需要注意的是在一个方法中调用另外一个方法，需要使用关键字$this。代码执行结果是：

```
这个数字的平方为: 64
```


❑ **protected**

protected 作用域表示方法是一个受保护的方法。例如，在获取某个员工信息之前，可能需要对员工信息进行校验，将员工信息作为参数传递到类的实例化方法（即构造函数）中，然后使用 verify_ein()方法验证语法是否正确。因为这个方法只用于类中的其他方法，对于 staff 派生的子类也可能有用，所以应声明为 protected，如代码 3.16 所示。

代码 3.16　使用 protected 关键字

```
<?php
   class staff{
       private $ein;
       function __construct($ein){
            if($this->verify_ein($ein)){
                echo "EIN verified Finish";
              }
         }
       protected function verify_ein($ein){
             return TRUE;
         }
    }
  $employee=new staff("123-45-6789");
?>
```

如果试图从类外部调用 protected 方法，将会出现致命性错误，protected 作用域对声明的方法具有保护作用。

❑ **abstract**

abstract 作用域可以用在类和方法的前面。通常一个方法被声明为抽象方法时，类的本身并没有实现该方法，一般是由类的子类来实现该抽象方法。如果一个类中包含了抽象方法，类本身也就成为了一个抽象类，该类的前面需要带有关键字 abstract。抽象类或者抽象方法定义了某些方面的共性，即一般性。抽象类主要实现了某些功能的命名约定。抽象方法的语法格式如下所示：

```
abstract function methodName()
```

抽象方法的应用示例如下所示：

```
abstract class Example{
   abstract function Abstr1();
   abstract function Abstr2();
```

```
    abstract function Abstr3();
}
```

在上述代码中创建了一个抽象类 Example，然后在其子类中实现这些抽象方法。

❑ **final**

final 作用域可以用在类和方法的前面，这些成员统称为 final 成员。当在一个方法的前面加上 final 作用域时，表示方法不可以被重写，即在该类的子类中，只允许调用，不允许重新设置该方法的功能。final 方法的语法格式为：

```
class Example{
      ...
     final function getValue(){}
      ...
}
```

3.3　构造函数和析构函数

在应用程序中，实例化类时需要为类中的字段赋值，此时需要在类中创建构造函数（constructor）来实现。类的对象在执行过程中，可能需要加载多个资源，如文件对象，数据库对象等。程序执行完毕后，如果不释放这些资源就会永远占有，其他的程序不能使用这些占有资源。这时，需要通过析构函数（destructor）来清除字段或者对象占有的资源。本节将主要介绍构造函数和析构函数。

3.3.1　构造函数

构造函数主要是在定义对象实例化时，自动执行的代码。构造函数可以接受参数，能够在创建对象时赋值给特定对象的字段，还可以调用类的方法、函数和其他构造函数，包括父类的构造函数。在 PHP 4 中，当函数与类同名时，这个函数将成为该类的构造函数。在 PHP 5 中，构造函数被统一命名为__construct。如果在一个类中声明一个函数，命名为__construct，这个函数将被当成是一个构造函数并在建立一个对象实例时被执行。严格地讲，“__”是两个下划线。就像其他任何函数一样，构造函数可以有参数或者默认值。可以定义一个类来建立一个对象并将其属性全放在一个语句中，构造函数的语法格式如下所示：

```
function __construct([arg1,arg2,...,argn]){
  方法体
}
```

PHP 可以通过名称__construct 来识别构造函数。下面创建一个示例，具体说明构造函数的应用，如代码 3.17 所示。

代码 3.17　创建构造函数

```
<?php
class Car
{
    private $carType;
    private $number;
    private $weight;
    function __construct($number)
    {
        $this->GetCarType();
        $this->GetWeight();
        $this->GetNumber($number);
    }
    function GetNumber($number)
    {
        $this->number = $number;
        echo "该车的车牌号是：".$this->number."<br/>";
    }
    function GetWeight()
    {
        $this->weight = "10 吨";
        echo "该车的重量是：".$this->weight."<br/>";
    }
    function GetCarType()
    {
        $this->carType ="奥迪 A8";
        echo "该车型号是：".$this->carType."<br/>";
    }
}
$c = new Car("京 A GH658");
?>
```

在代码 3.17 中，创建了一个 Car 类，在该类中首先定义了 3 个字段，并创建了一个构造函数和 3 个方法，该构造函数调用了在类中预定义好的方法。该段代码执行结果如下所示：

```
该车型号是：奥迪 A8
该车的重量是：10 吨
该车的车牌号是：京 A GH658
```

在构造函数中不仅可以调用类本身的方法，还可以调用其他类的构造函数，调用其他类的构造函数的语法格式为：

```
classname::__construct();
```

下面创建一个调用其他类构造函数的示例。首先创建一个类，在该类中创建一个构造函数，接下来再创建一个类，并在新类中创建一个构造函数，然后在该构造函数中调用第一个

类的构造函数。同样，在子类中也可以调用父类的构造函数，对于该知识点的介绍会在后面的章节提到，示例如代码 3.18 所示。

代码 3.18　调用其他类构造函数

```
<?php
class HelloClass {
   function __construct($x)
   {
      echo $x;
   }
 }
class WorldClass
 {
   function __construct($y)
   {
      HelloClass::__construct("Hello  ");
      echo $y;
   }
 }
$w = new WorldClass("World~~!");
?>
```

代码 3.18 的执行结果是：

```
Hello  World~~!
```

3.3.2　析构函数

在 PHP 4 以前的版本中，尽管脚本执行程序结束后对象会自动撤销，但是不能消除过程。随着 PHP 5 的出现，这种限制就不存在了，主要是因为在 PHP 5 中引入了析构函数。析构函数可以用来记录调试信息，关闭数据库连接等一些清除收尾的工作。当指向这个对象的最后一个引用被销毁时，析构方法被调用，调用完成后释放内存，并且析构函数不接受任何参数。

创建析构函数与创建其他方法一样，但是析构函数必须命名为__destruct()。PHP 调用析构函数来将一个对象从内存中销毁。默认 PHP 仅仅释放对象属性所占用的内存并销毁对象相关的资源。析构函数允许在使用一个对象之后执行任意代码来清除内存。下面创建一个示例，如代码 3.19 所示。

代码 3.19　创建析构函数

```
<?php
class FirstDestruct {
  var $x;
  function __construct($x) {
    echo "类的构造函数，创建类对象时，必须有参数<br/>";
```

```
    }
    function __destruct() {
      print("释放内存中的所有对象");
    }
  }
  $o1 = new FirstDestruct (8);
  ?>
```

在代码 3.19 中，创建了一个类 FirstDestruct，该类重写了构造函数和析构函数，脚本执行结束时会执行析构函数，撤销内存中的所有对象。如果实例化的类和实例化创建的信息留在内存中，就不需要显示地声明析构函数。但是如果实例化时创建了不容易丢失的数据，就应当在对象撤销时撤销这些数据，为此需要创建一个定制的析构函数，即重写析构函数__destruct()。执行代码 3.19，结果如下所示：

```
类的构造函数，创建类对象时，必须有参数
释放内存中的所有对象
```

3.4 新增 OOP 特性

在 PHP 5 中增加了许多 OOP 特性，本节主要讲解这些新增的 OOP 特性，例如类型提示、instanceof 关键字和自动加载对象等。

3.4.1 类型提示

类型提示是 PHP 5 中的一种新特征，类型提示能够将函数参数强制转换成特定类型的对象，可以简单地通过在函数参数的前面加上类型名强制转换对象类型。下面创建一个使用类型提示的示例，如代码 3.20 所示。

代码 3.20 使用类型提示

```
<?php
class MyClass
  {
    public $var = '这里用了类型提示';
  }
function MyFunction(MyClass $foo)
{
echo $foo->var;
}
$myclass = new MyClass;
MyFunction($myclass);
?>
```

在代码 3.20 中，创建了一个类 MyClass，该类中存在一个变量 var。之后创建了一个函数 MyFunction，该函数的参数是 MyClass 类的一个对象 foo，此时 foo 为形参。在函数体中，直接输出了对象 foo 的成员变量 var 的值。接下来创建了该类的实例化对象$myclass，并以此作为参数传递给函数 MyFunction。类型提示只能够是对象和数组（从 PHP 5.1 开始）类型，传统的类型提示不支持整型和字符串型。

3.4.2　静态类成员

类的静态成员与一般的类成员不同，静态成员与对象的实例无关，只与类本身有关。类的静态成员用来实现类要封装的功能和数据，但是不包括特定对象的功能和数据，类的静态成员包括静态方法和静态属性。本节将详细介绍如何创建静态成员和使用静态成员的方法。

声明一个静态成员，只需要在方法前面或者字段前面加上 static 关键字即可。一个类的静态成员会被该类所有的实例化对象共享，任何一个实例化对象对静态成员的修改都会映射到静态成员中。根据这个性质，可以把静态成员看作一个全局变量，静态声明必须在可见性声明之后。

静态属性是包含在类中要封装的数据，可以由所有类的实例化对象共享。实际上，除了属于一个固定类并限制访问方式外，类的静态属性非常类似于函数的全局变量。静态属性不能通过箭头操作符“->”访问。

静态方法实现类需要封装的功能，与特定的对象无关。静态方法非常类似于全局函数。静态方法可以完全访问类的属性，也可以由对象的实例来访问。由于静态方法可以由非对象实例调用，因此变量$this 不可以在声明为静态的方法中使用。事实上 static 方法调用形式在编译时被确定。下面创建一个示例，具体介绍静态成员如何使用，如代码 3.21 所示。

代码 3.21　使用静态成员

```
<?php
class Counter
{
private static $count = 0;
const VERSION = 2.0;
function __construct()
{
self::$count++;
}
function __destruct()
{
self::$count--;
}
static function getCount()
{
return self::$count;
}
```

```
};
$c = new Counter();//创建一个实例,执行构造函数
print("输出的值为: ".Counter::getCount() . "<br/>");//输出 1
print("类的版本号: " . Counter::VERSION . "<br/>");//输出类的版本属性
?>
```

在代码 3.21 中，创建了一个私有的静态成员$count，一个常量 VERSION 并赋值为 2。在类的构造函数中，通过 self 关键字，调用静态成员$count，在这里不能用$this 来引用它，但可以用 self 或其他有效的命名表达。然后在该类的构造函数中，使用静态成员执行自动加 1，表示每创建一个实例化对象，该值就加 1。在类的析构函数中，使$count 变量的值减 1。在静态方法 getCount()中，返回静态变量$count 的值。创建类的对象，并调用相应的方法。运行程序，执行结果如下所示：

```
输出的值为: 1
类的版本号: 2.0
```

3.4.3 instanceof 关键字

PHP 5 另一个新成员是 instanceof 关键字，instanceof 关键字用来测试一个对象是否是一个类的实例，或者是一个派生类的实例，或者是某个接口的实例，并进行相应的操作。下面创建一个示例，具体讲解 instanceof 关键字如何使用，如代码 3.22 所示。

代码 3.22 使用 instanceof 关键字

```
<?php
class myClass { };
$a = new myClass;
if ($a instanceof myClass)
{
   echo "对象\$a 是 myClass 类的实例化对象";
}
else
{
   echo "对象\$a 不是 myClass 类的实例化对象";
}
?>
```

在代码 3.22 中，首先创建了一个 myClass 类，该类没有任何一个成员。然后创建了一个 myClass 类的对象$a，并且使用 if 语句判断对象$a 是否是 myClass 类的实例化对象，如果为 true 则在页面上输出“对象$a 是 myClass 类的实例化对象”，否则输出“对象$a 不是 myClass 类的实例化对象”。instanceof 关键字可以同时处理多个对象，例如，当程序员要重复地调用某个函数，但希望根据对象类型调整函数的行为时，可以使用 case 语句和 instanceof 关键字实现。

3.4.4　自动加载对象

创建一个大型的 PHP 项目，可能需要创建大量的类库，从而达到功能的相互分离，以及代码的重复使用。在一个 PHP 页面中，要使用这些类库或者函数，需要通过 require_once()语句来实现。例如，如果 PHP 页面中需要使用 1 个类，就需要在前面插入语句：

```
require_once("myweb/student.class.php");
```

在一个 PHP 页面中，可能需要使用多个 require_once()语句。在编写这些语句代码时，假如是一行一行地编写，会变得非常繁琐。为消除这个额外任务，PHP 5.0 中引入了自动加载对象的概念。自动加载需要通过 autoload()函数实现。

当尝试使用一个未定义的类时，PHP 会报告一个致命错误。解决方法就是添加一个类，可以用 include 包含一个文件，但需要知道要调用哪个类。PHP 提供了类的自动加载功能，可以节省编程的时间。当尝试使用一个 PHP 没有组织到的类时，PHP 会寻找一个__autoload()的全局函数。如果存在这个函数，PHP 会用一个参数来调用这个函数，参数即类的名称，其使用示例如代码 3.23 所示。

代码 3.23　使用 include

```
<?php
function __autoload($className) {
   include_once  $className . ".php";
//require_once("classes/$class.class.php");
}
$object = new ClassName;
?>
```

在上述代码中，函数中语句表示加载指定 PHP 页面。如果定义了这个函数，就不需要再使用 require_once()语句或者 include_once 语句。

3.5　类/对象函数

在 PHP 5.0 中，提供了一些内置函数，用来帮助开发人员管理和使用类库，例如 class_name、get_class、is_a、is_subclass_of 和 method_exists 等，本节将详细介绍部分内置函数。

3.5.1　class_exits()函数

在 PHP 5 中执行脚本上下文时，如果有 class_name 所指的类存在，则 class_exists()函数返回 true，否则返回 false。该函数语法格式如下所示：

```
bool class_exists ( string $class_name [, bool $autoload] )
```

下面使用 class_exists()函数创建一个示例，如代码 3.24 所示。

代码 3.24　使用 class_exists()函数

```
<?php
 class ClassName
 {
    function __construct()
    {
        echo "存在 ClassName 类";
    }
 }
 if (class_exists("ClassName"))
 {
    $s = new ClassName();
 }
 else
 {
    echo "不存在 ClassName 类";
 }
?>
```

在代码 3.24 中，首先创建了 ClassName 类，然后使用 class_exists()函数判断该类是否存在，如果存在实例化 ClassName 类，否则在页面上打印“不存在 ClassName 类”。

3.5.2　get_class()函数

get_class()函数返回对象所属类的类名。在 PHP 扩展库中定义的类返回其原始定义的名字，而在 PHP 4 中 get_class()函数返回用户定义的类名的小写形式，如同扩展库中的类名一样。自 PHP 5 起，如果在对象的方法中调用，则对象为可选项。get_class()函数的语法格式如下所示：

```
string get_class(object object)
```

下面创建一个使用 get_class()函数的示例，如代码 3.25 所示。

代码 3.25　使用 get_class()函数

```
<?php
class ClassName
 {
   function name()
   {
       echo "类名是: ".get_class($this)."<br/>";
   }
 }
$bar = new ClassName();
```

```
echo "对象的名称为: ".get_class($bar)."<br/>";
$bar->name();
?>
```

在代码 3.25 中，首先创建了一个 ClassName 类，并在该类中定义了一个 name()方法，然后实例化该类，最后调用该类的 name()方法。执行代码 3.25，结果如下所示：

```
对象的名称为: ClassName
类名是: ClassName
```

3.5.3　get_class_methods()函数

get_class_methods()函数返回一个数组，其中包含类中定义的所有方法名，如果出错返回 null，该函数语法格式如下所示：

```
array get_class_methods ( mixed $class_name )
```

下面创建一个使用 get_class_methods()函数的示例，如代码 3.26 所示。

代码 3.26　使用 get_class_methods()函数

```
<?php
class ClassName
{
    function MethodOne()
    {
        echo "方法一";
    }
    function MethodTwo()
    {
        echo "方法二";
    }
}
$classMethod = get_class_methods('ClassName');
$classMethod = get_class_methods(new ClassName());
foreach ($classMethod as $className)
{
    echo $className."<br/>";
}
?>
```

运行程序，执行结果如下所示：

```
MethodOne
MethodTwo
```

3.5.4 get_class_vars()函数

该函数返回由类的默认公有属性组成的关联数组，此数组的元素以 varname =>value 的形式存在，其中还包含类中定义的所有字段名及相应的值，该函数语法格式如下所示：

```
array get_class_vars ( string $class_name )
```

下面创建一个使用 get_class_vars()函数的示例，如代码 3.27 所示。

代码 3.27 使用 get_class_vars()函数

```
<?php
class ClassName
{
    public  $var1; // 此变量没有默认值
    public $var2 = "xyz";
    public $var3 = 100;
    private $var4;
   function __construct()
    {
       $this->var1 = "foo";
       $this->var2 = "bar";
    }
}
$myclass = new ClassName();
$classVars = get_class_vars(get_class($myclass));
foreach ($classVars as $varName => $value)
{
   echo "$varName : $value<br/>";
}
?>
```

执行代码 3.27，结果如下所示：

```
var1 :
var2 : xyz
var3 : 100
```

3.5.5 get_declared_classes()函数

get_declared_classes()函数返回由当前脚本中已定义类的名字组成的数组，该数组包括当前执行脚本中定义的所有类名，根据 PHP 发行包的配置不同，该函数输出会有所区别。该函数的语法格式如下所示：

```
array get_declared_classes ( void )
```

下面创建一个使用 get_declared_classes()函数的示例，如代码 3.28 所示。

代码 3.28　使用 get_declared_classes()函数

```
<?php
print_r(get_declared_classes());
?>
```

3.5.6　get_object_vars()函数

get_object_vars()函数返回由指定的对象中定义的属性组成的关联数组，其中包含对象可用的已定义字段及其相应的值，该函数语法格式如下所示：

```
array get_object_vars(object object)
```

下面创建一个使用 get_object_vars ()函数的示例，如代码 3.29 所示。

代码 3.29　使用 get_object_vars()函数

```
<?php
class ClassName {
    public $x, $y;
    public $label;
    function ClassName ($x, $y)
    {
        $this->x = $x;
        $this->y = $y;
    }
    function setLabel($label)
    {
        $this->label = $label;
    }
    function getPoint()
    {
        return array("x" => $this->x,
                     "y" => $this->y,
                     "label" => $this->label);
    }
}
$p = new ClassName(1.233, 3.445);
print_r(get_object_vars($p));
$p->setLabel("point #1");
print_r(get_object_vars($p));
?>
```

该段代码执行结果如下所示：

```
Array ( [x] => 1.233 [y] => 3.445 [label] => ) Array ( [x] => 1.233 [y] => 3.445
[label] => point #1 )
```

3.5.7 method_exists()函数

如果方法名所指的方法在对象所指的类中已定义，则返回 true，否则返回 false，该函数语法格式如下所示：

```
Boolean method_exists(object object,string className)
```

下面创建一个示例，具体讲解该函数如何使用，代码如下所示：

```
<?php
$directory = new Directory('.');
var_dump(method_exists($directory,'read'));
?>
```

3.5.8 interface_exists()函数

interface_exists()函数确定一个接口是否存在，如果存在返回 true，否则返回 false，该函数语法格式如下所示：

```
Boolean interface_exists(string interface_name[,Boolean autoload])
```

由于该函数比较简单，在此就不再列举示例。

第4章 高级 OOP 特性


内容摘要 | Abstract

在第 3 章中讲解了 PHP 面向对象编程的基础，及面向对象的特性、类成员、构造函数和析构函数等。本章将在上一章的基础上进一步介绍一些更高级的 PHP 面向对象特性，例如对象克隆、继承、接口和反射等。


学习目标 | Objective

- 了解 PHP 不支持哪些 OOP 高级特性
- 灵活运用 PHP 对象之间的克隆
- 掌握__clone()方法的使用
- 灵活掌握类之间的继承
- 掌握如何创建接口和使用接口
- 了解在 PHP 中接口之间的继承关系
- 掌握在 PHP 中抽象类的创建和使用
- 了解 PHP 的反射

4.1 PHP 不支持的高级 OOP 特性

在使用 PHP 语言编程之前，如果读者使用过其他编程语言，会发现 PHP 并不完全包括其他完全面向对象语言的特性，例如命名空间、方法重载和操作符重载等。本节主要介绍 PHP 不支持的高级 OOP 特性。

- **命名空间** 在 PHP 5.0 的初期计划中，本来准备将命名空间作为 PHP 的一个特性，但是后来去除了对命名空间的支持。到目前为止还不清楚后续 PHP 版本会不会将命名空间集成到以后的版本中。
- **永久对象** 在 OOP 中，永久对象是可以在多个应用的引用中保持状态和功能的对象，这意味着拥有将对象保存到一个文件或数据库中的能力，而且可以在以后装入对象，这就是所谓的序列化机制。PHP 拥有序列化方法，可以通过对象进行调用，序列化方法可以通过返回对象的字符串来表示。然而，序列化只保存了对象的成员数据而不包括方法。
- **多重继承** 继承的出现一方面节省了代码的编写，另一方面使类具有了一个很好的层

次结构。在 PHP 中支持单继承，不支持多重继承。多重继承是指类不仅可以继承于一个类，并且能同时继承于多个类，也可以说一个类可以派生于多个类。当程序需要时，则需要预先设计好多个接口，让类继承。

- **方法重载**　PHP 不支持通过方法重载实现多态，根据 Zend 网站的讨论，PHP 可能永远都不会支持方法重载。方法重载，实际是类多态的一种表现形式，即多个方法具有相同的方法名称，但是具有不同的参数个数和类型。通过名称调用方法，根据参数决定执行的是哪个方法。
- **操作符重载**　PHP 目前还不支持根据修改数据的类型为操作符赋予新的含义，根据 Zend 网站的讨论，PHP 中实现该 OOP 特性的可能性也不会很大。

4.2　对象克隆

由于在 PHP 的早期版本中将对象视为一种数据类型，因此许多面向对象的特性无法使用，例如当编写程序时，需要一个现存对象的副本。但是在 PHP 中可以克隆一个现存对象，也可以说是将一个对象的状态复制到另一个对象中，这是因为在 PHP 5 中提供了一种克隆（clone）对象的显式方法。本节主要讲解对象克隆。

4.2.1　克隆

通过完全地复制属性创建一个对象的备份通常不是想要的行为。有个例子可以很好地说明确实需要复制对象。例如一个 GTK 窗口的对象 a，a 持有其所需要的全部资源。当复制这个 GTK 窗口到对象 b 时，程序中更希望 b 持有新的资源对象。再举个例子：对象 a 包含了一个对象 c，当程序员将对象 a 复制到对象 b 时，可能更希望对象 b 包含一个新对象 c 的复制，而不是一个对象 c 的引用。在 PHP 中这些可以使用关键字 clone 实现，下面使用该关键字创建一个示例，具体讲解对象克隆，如代码 4.1 所示。

代码 4.1　使用 clone 关键字

```
<?php
class Student {
  private $stuid;
  private $stuname;
  function SetStuID($stuid)
   {
     $this->stuid = $stuid;
    }
  function GetStuID()
   {
     return $this->stuid;
    }
```

```
      function SetStuName($stuname)
       {
          $this->stuname = $stuname;
        }
      function GetStuName()
       {
           return $this->stuname;
       }
   }
   $stu1 = new Student();
   $stu1 ->SetStuID("200805006");
   $stu1 ->SetStuName("唐晓阳");
   $stu2 = clone $stu1;
   $stu2->SetStuID("200805008");
   echo "stu1 学生编号: ".$stu1->GetStuID()."<br/>";
   echo "stu1 学生姓名: ".$stu1 ->GetStuName()."<br/>";
   echo "stu2 学生编号: ".$stu2 ->GetStuID()."<br/>";
   echo "stu2 学生姓名: ".$stu2->GetStuName()."<br/>";
   ?>
```

将上述代码命名为 Demo-Clone.php 保存到 F:\MyWeb\Apache\htdocs 目录下，打开 IE 浏览器，在地址栏中输入 http://localhost/Demo-Clone.php，单击【转到】按钮将会显示如图 4-1 所示的窗口。

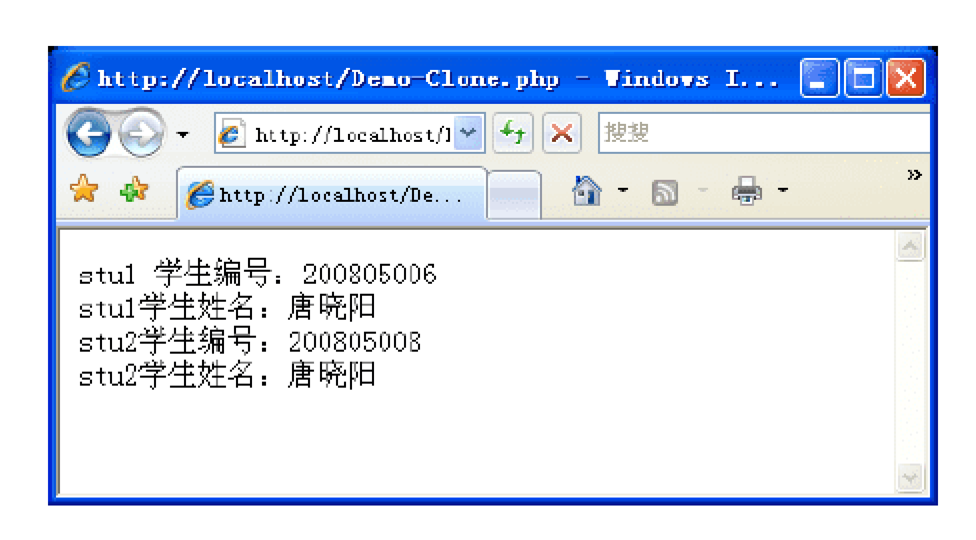


图 4-1　对象克隆

从代码 4.1 中可以看出，$stu2 变成一个 Student 类型的对象，继承了$stu1 的成员值，并且可以为对象$stu2 成员重新赋值。

4.2.2　__clone()方法

在对象的克隆期间，有时不希望将所有的属性克隆过来，希望在某些局部位置做一点改动，如一个字段的值。这时可以在克隆的类中定义一个方法__clone()，从而调整对象的克隆行为，即在克隆过程中，将一些局部的代码设置成自己需要的代码。在将现有的对象成员复制到目标对象之后，会执行__clone()方法指定的操作。

下面创建一个实例，演示__clone()方法的使用，具体如代码 4.2 所示。

代码 4.2　__clone()方法

```
<?php
class MyClone {
  public static $uid = 1;
   function __construct() {
       $this->uid = self::$uid++;
   }
   function __clone() {
       $this->address = "北京";
       $this->uid = self::$uid++;
   }
}
$obj = new MyClone();
$obj->username = "唐晓阳";
$obj->sno="851225";
$obj->address = "河南、郑州";
print $obj->uid . "<br/>";
$obj cloned = clone $obj;
print $obj cloned->uid . "<br/>";
print $obj cloned->username . "<br/>";
print $obj cloned->sno . "<br/>";
print $obj_cloned->address . "<br/>";
?>
```

保存代码 4.2，运行程序，页面效果如图 4-2 所示。

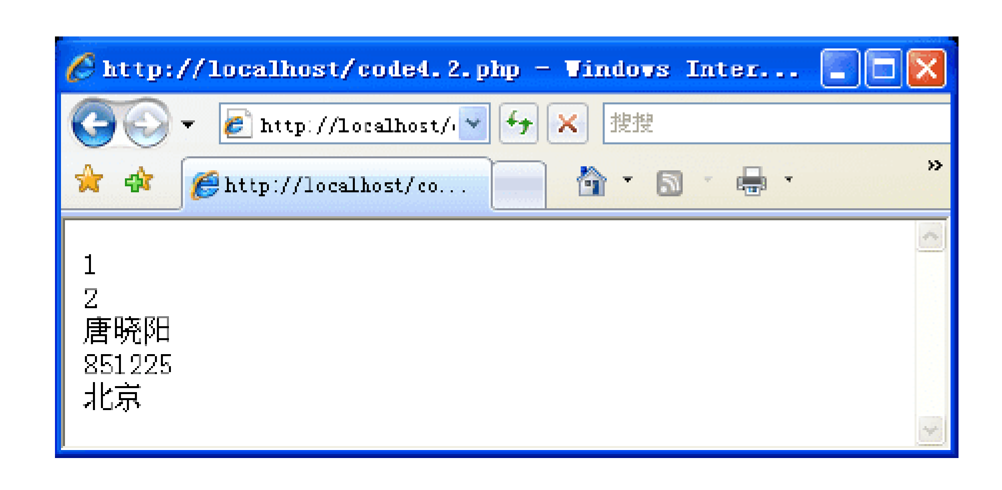


图 4-2　__clone 案例

在代码 4.2 中，创建了一个类 MyClone，并为该类创建一个静态变量，一个构造函数和一个__clone()方法。在构造函数中，设置静态变量$uid 的值，每调用一次$uid 的值就增加一次。在__clone()方法中，设置在进行对象克隆后，还需要调整的代码，如地址信息为“北京”，变量$uid 的值在原来的基础上加 1。使用语句$obj =new MyClone()创建一个对象$obj，并利用该对象设置变量$username 和$address 的值，并输出变量$uid 的值。使用语句$obj_cloned = clone $obj 产生一个新的克隆对象，这时该对象具有$obj 的属性和方法，然后输出克隆的对象$obj_cloned 此时所具有的变量的值。其中变量$uid 的值变化了，变量$address 的值不再是前面设定的“河南、郑州”，而是在__clone()方法中设置的“北京”信息。从本实例中可以得出，__clone()方法可以修改克隆对象中的一些代码。

4.3　继承

继承是面向对象程序设计中一个很重要的特性，是关于一个类怎样从另一个类中共享特性和行为的术语。PHP 中的继承类称为派生类或子类，被继承类称为基类或父类。派生类能够继承基类的所有操作、属性、特性、事件以及这些内容的实现方法，但继承得到的成员也受访问限制，即使派生类继承得到基类成员，也可能无法访问。继承的产生使类和类之间有了相应的层次结构，使类的管理更加清晰，并且提高了代码的可重用性，比如一些功能可以通过继承来获得。在 PHP 中，可以实现一个类继承另外一个类，但是不支持多重继承，即继承多个类。本节将详细介绍继承的相关知识。

4.3.1　类继承

在 PHP 中，类继承通过 extends 关键字实现，并且类继承属于单继承，也就是说一个类只能有一个基类。但是一个类可以被多个子类继承，继承的语法格式如下所示：

```
class class1 extends class2
{
   //类成员
}
```

在上述代码中，class1 表示子类，extends 表示类的继承符，class2 表示父类。子类不但可以拥有父类的成员，例如方法和字段，还可以拥有本身新增的方法，但是子类不能拥有父类的私有成员。下面创建一个示例，具体讲解类的继承，如代码 4.3 所示。

代码 4.3　类继承

```
<?php
class Student
{
   private $name;
   public $ID = "20080608";
   function SetName($name)
   {
       if($name == "")
       {
          echo "请设置学生姓名";
       }
       else
       {
          $this->name = $name;
       }
   }
```

```
    function GetName()
    {
       return "学生姓名是: ".$this->name."<br/>";
    }
    function printStudent()
    {
       echo "学生编号是: ".$this->ID."<br/>";
    }
}
class ExtendsStudent extends Student
{
    function printWorld()
    {
        if($this->ID=="20080608")
        {
            echo  "子类 ExtendsStudent 成功地继承于父类 Student<br/>";
        }
    }
}
$class1 = new Student();
$class1->SetName("唐晓阳");
echo $class1->GetName()."<hr/>";
$class2 = new ExtendsStudent();
echo "执行子类程序:<br/><hr/>";
echo "子类对象调用父类中的函数: <br/>";
$class2 ->printStudent();
echo "<hr/>";
echo "子类对象执行子类中的函数: <br/>";
$class2 ->printWorld();
?>
```

保存代码，并重命名文件为 Demo-Extends.php，然后保存到指定位置。打开 IE 浏览器，在地址栏中输入http://localhost/Demo-Extends.php，单击【转到】按钮，页面显示效果如图 4-3 所示。

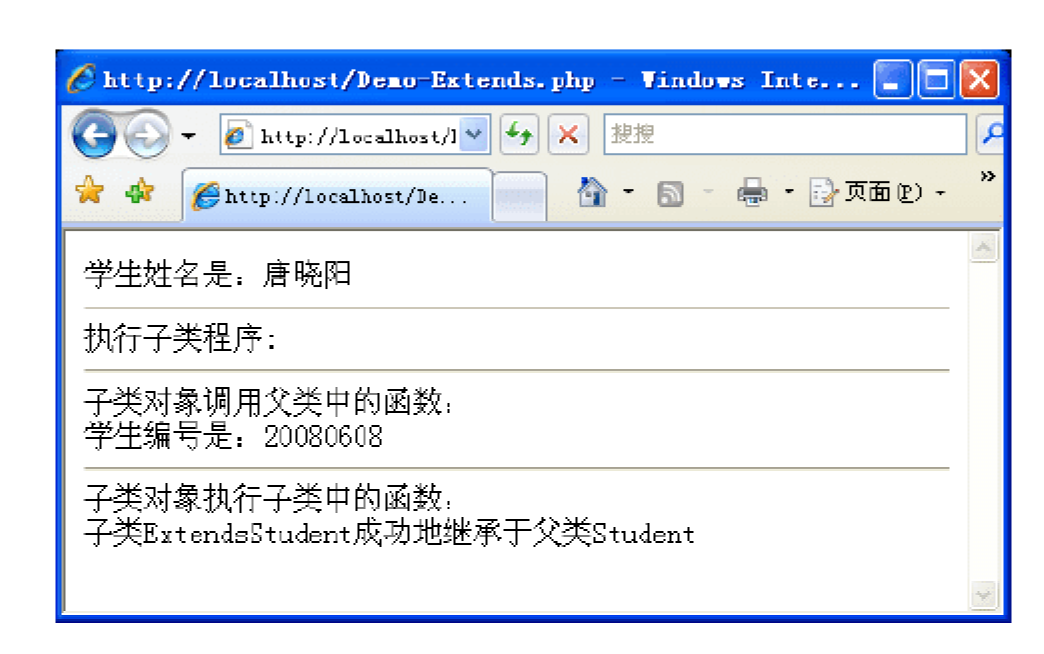


图 4-3　类继承页面效果图

在代码 4.3 中，首先创建了一个 Student 类，然后在类中创建了一个私有变量、一个公共变量、一个属性和一个方法，该类中的方法输出学生编号信息，最后为该类创建了一个子类，并在子类中创建一个方法。创建好类后，在 PHP 应用程序中实例化基类对象$class1 和子类对象$class2，并利用对象调用相关方法和属性。由该示例知，一个子类可以继承父类中的公共成员，但却不能访问基类中的私有成员。

4.3.2　继承和构造函数

在上一节中讲解了类的继承，子类可以从基类中继承所有公共成员，当然这也包括构造函数。在子类中运用基类中的构造函数与运用其他成员不同，不同之处在于子类可以显式调用基类中的构造函数，也可以隐式调用基类中的构造函数。本节主要讲解继承和构造函数。

如果父类中有构造函数，并且子类中没有构造函数，那么子类在实例化时执行父类的构造函数。下面创建一个示例，如代码 4.4 所示。

代码 4.4　继承与构造函数

```
<?php
 class Class1
 {
    function __construct()
    {
        echo "该构造函数为基类构造函数";
    }
 }
 class Class2 extends Class1
 {
    function Write()
    {
        echo "<br/>该方法为子类方法";
    }
 }
 echo "查看当实例化子类时是否调用父类构造函数:<br/>";
 $class = new Class2();
 $class->Write();
?>
```

在代码 4.4 中，首先创建了一个基类，并且在该类中创建一个构造函数，然后为该类创建了一个子类，在子类中无构造函数，最后实例化子类，查看子类是否调用父类中的构造函数。保存代码 4.4，执行结果如下所示：

```
查看当实例化子类时是否调用父类构造函数:
该构造函数为基类构造函数
该方法为子类方法
```

在上一章中介绍了一个类可以调用另外一个类的构造函数，即使这两个类不是基类与子类关系也可以。非继承关系调用构造函数称为显式调用。如果一个子类继承了一个父类，子类只需要使用关键字 parent 就可以直接调用父类构造函数，称为隐式调用。下面创建一个示例，具体讲解子类调用父类构造函数，如代码 4.5 所示。

代码 4.5　使用关键字 parent 调用基类构造函数

```
<?php
class BaseClass {
   function __construct()
   {
       print "父类的构造函数<br/>";
   }
}
class SubClass extends BaseClass {
   function __construct()
   {
        echo "显式调用基类构造函数: ";
       BaseClass::__construct();
       echo "隐式调用基类构造函数: ";
       parent::__construct();
       echo "执行子类本身构造函数: ";
       print "子类的构造函数<br/>";
   }
}
echo "<h4>实例化父类,执行结果如下所示:</h4>";
$obj = new BaseClass();
echo "<br/><br/>";
echo "<h4>实例化子类，执行结果如下所示:</h4>";
$obj = new SubClass();
?>
```

保存代码 4.5，并重命名为 Demo-jichengConstruct，然后保存到指定的位置。打开 IE 浏览器，在地址栏中输入 http://localhost/Demo-jichengConstruct.php，单击【转到】按钮，页面显示效果如图 4-4 所示。

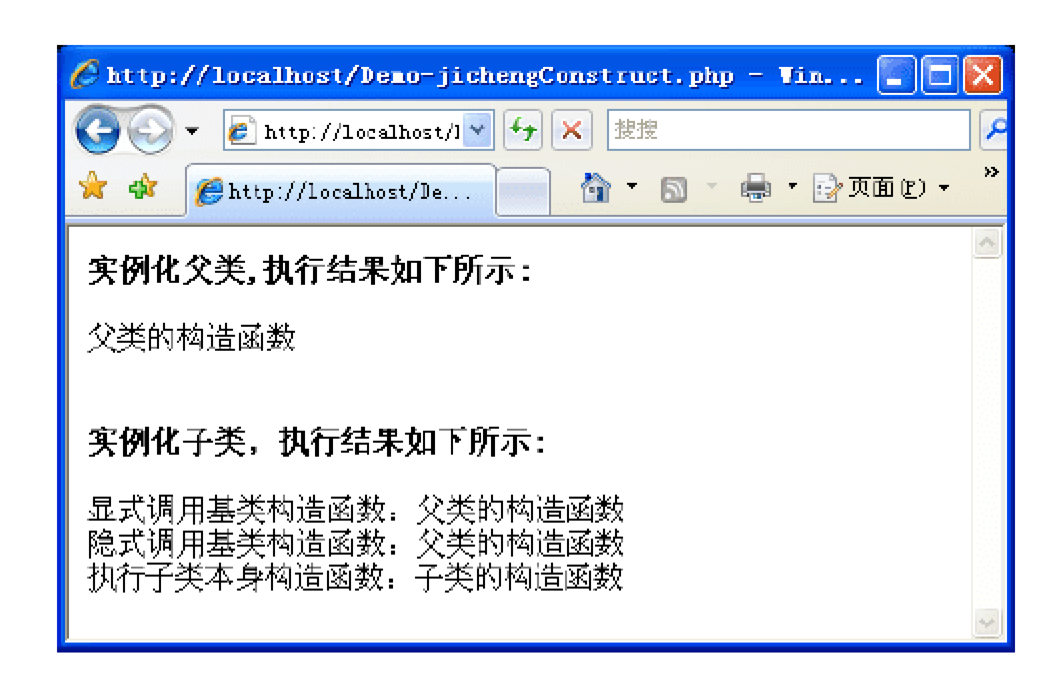


图 4-4　构造函数

4.4　接口

在编程术语中，接口类似于类，但是接口的成员都没有执行方式。接口只是方法和属性的组合而已，没有构造函数，也没有字段，并且还不能被实例化。接口只包含成员的签名，也可以说是对类的一种规范。本节将会详细地介绍接口的创建和使用。

4.4.1　实现一个接口

接口允许创建一个指定类方法的执行代码，而不必说明这些方法如何被操作，也可以说是类的一种规范。接口作为一个标准类，包含一些方法，但是这些方法都没有方法体。在接口中所有的方法必须声明为 public，这是接口的特性。创建一个接口需要使用关键字 interface，创建接口的语法格式如下所示：

```
interface Myinterface{
  const name1;
    ...
  const namen;
  function methodName1();
   ...
  function methoodNamen();
}
```

在上述代码中，interface 表示创建一个接口，Myinterface 表示接口的名称。name1 等表示在接口中声明的字段，是常量。methodName1 表示方法的名称，该方法没有方法体。

接口的使用意义在于定义了一系列的标准，让其他类去实现。如果创建了一个接口，而没有相关的类去实现，该接口是毫无意义的。接口需要通过继承来实现其价值。一个类继承于接口需要使用 implements 关键字，该关键字表示实现一个接口。在接口中所有的方法必须在实现类的内部实现，疏忽这些将导致一个致命错误。类可以实现多个接口，通过逗号隔开。下面创建一个实现接口的示例，如代码 4.6 所示。

代码 4.6　实现一个接口

```
<?php
interface IMyInterface
{
    public function GetName();
    public function Achieve($num);
}
class AchieveInterface implements IMyInterface
{
    public function GetName()
```

```
        {
            echo "请你详细登记新生姓名，以及其他详细信息<br/>";
        }
        public function Achieve($num)
        {
            echo "班级共有".$num."名学生";
        }
    }
    $ache = new AchieveInterface();
    $ache->GetName();
    $ache->Achieve(50);
    ?>
```

在代码 4.6 中，首先定义了一个接口，然后使用类 AchieveInterface 类实现该接口。AchieveInterface 类实现了接口中所有的方法，如果不实现接口中的全部方法，则会出现错误。执行代码 4.6，结果如下所示：

```
请你详细登记新生姓名，以及其他详细信息
班级共有 50 名学生
```

4.4.2 实现多个接口

一个类不仅可以实现一个接口，也可以同时实现多个接口。当实现多个接口时，接口之间用逗号隔开。下面创建一个示例，说明类如何实现多个接口，具体如代码 4.7 所示。

代码 4.7 实现多个接口

```
<?php
interface IShowNameable {
  function myname();
}
interface IOtherNameable {
  function othername();
}
class showname implements IShowNameable ,IOtherNameable {
  function myname() {
    echo "姓名：唐晓阳";
  }
 function othername() {
    echo "姓名：裴亚敏";
 }
}
$show=new showname();
$show->myname();
echo "<br/>";
```

```
$show->othername();
?>
```

保存并执行代码 4.7，页面效果如图 4-5 所示。

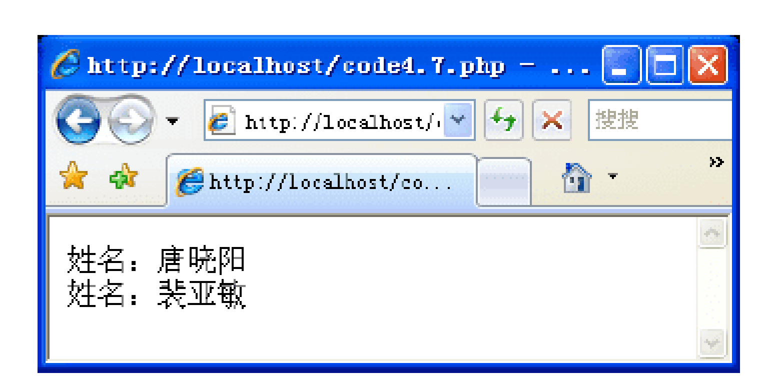


图 4-5　实现多个接口

在上述代码中，创建一个接口 IShowNameable，定义了一个方法 myname。又创建了另外一个接口 IOtherNameable 和一个方法 othername。创建了一个类 showname 继承了上面两个接口，并实现里面的方法。然后创建 showname 类的对象$show，利用该对象调用已实现的 myname()方法和 othername()方法。

接口实际上可以看作一个类，类和类之间可以继承，那么接口和接口之间是否可以发生关系？在 PHP 中，一个接口可以继承另外一个接口。创建一个实例，具体如代码 4.8 所示。

代码 4.8　实现接口继承

```
<?php
interface IShowNameable {
    public function myname();
}
interface IOtherNameable extends IShowNameable {
    public function othername();
}
class showname implements IOtherNameable {
    public function myname() {
        echo "EastJia";
    }
    public function othername() {
        echo "Joy";
    }
}
$show = new showname();
$show->myname();
echo "<br/>";
$show->othername();
?>
```

在代码 4.8 中，创建一个接口 IOtherNameable，并且继承接口 IShowNameable，那么 IOtherNameable 接口中就存在了两个方法 myname()方法和 othername()方法。

4.5 抽象类

有时基类并不与具体的事物相联系，而是只表达一种抽象的概念，为其派生类提供一个公共的界面，为此 PHP 中引入了抽象类的概念。抽象类不能实例化，只允许被继承，然后实例化子类，也可以将抽象类看成是子类的一个模板。并且抽象方法没有执行代码，必须在非抽象的派生类中重写。抽象方法就像是子类中一般方法的占位符，与一般方法不同，抽象方法没有任何代码。如果类包含抽象方法，那么该类也必须声明是抽象类。在 PHP 中，抽象类和抽象方法都用 abstract 关键字声明。

下面创建一个示例，具体讲解如何创建抽象类和抽象方法，以及如何实现抽象类中的抽象方法，如代码 4.9 所示。

代码 4.9　创建抽象类和抽象方法

```
<?php
abstract class AbstractClass
{
    abstract protected function getScore();
    abstract protected function getName($name);
    public function printOut()
    {
        print "分数是: ".$this->getScore()."<br/>";
    }
}
class Class1 extends AbstractClass
{
    protected function getScore()
    {
        return 80;
    }
    public function getName($name)
    {
        return "<br/>Class1 获取姓名: {$name}";
    }
}
class Class2 extends AbstractClass
{
    public function getScore()
    {
        return 99;
    }
    public function getName($name)
    {
```

```
        return"<br/>Class2 获取姓名: {$name}";
    }
}
$class1 = new Class1();
echo $class1->getName('裴亚敏')."的";
$class1->printOut();
$class2 = new Class2();
echo $class2->getName('唐晓阳') ."的";
$class2->printOut();
?>
```

保存并执行代码 4.9，页面效果如图 4-6 所示。

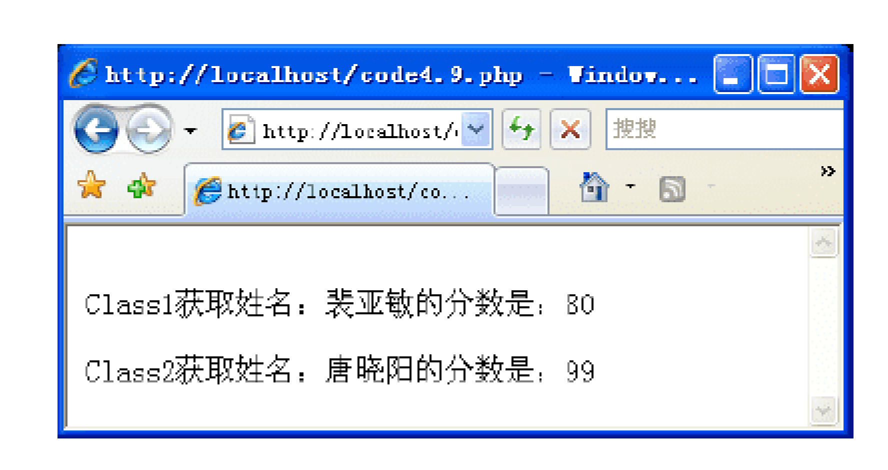


图 4-6　抽象类的实现

在本例的代码中，创建了一个抽象类 AbstractClass，该类的前面需要加上关键字 abstract 表明该类为抽象类。在 AbstractClass 中，定义了抽象方法 getScore()和带有参数的 getName()抽象方法，还创建了一个普通方法 printOut()，该方法输出一个字符串信息。创建另外两个类 Class1 和 Class2 继承上面的抽象类，这两个类分别实现了抽象类中的方法。抽象类不能被直接实例化，可以通过实例化抽象类的子类调用相应的方法。

4.6　反射

一个大型应用程序，肯定由很多类组成，并且每个类中可能包含大量成员。程序设计过程中可能需要查看某些方法的作用域范围，如果打开多个 PHP 页面会很不方便。在 PHP 中，提供了一些反射 API，不仅能够查看类和方法，还能够查看函数、接口和子类。反射不仅仅用来获取信息，还可以测试文档，生成一些帮助文件等。本节将一一介绍这些反射 API。

4.6.1　编写 ReflectionClass 类

ReflectionClass 类主要用于了解类的信息，例如该类具有哪些成员、类的名称、该类是否为抽象类和是否为最终类等。ReflectionClass 类继承接口 Reflector，该类封装和实现了一些相应的方法，用来获得要反射的类的基本信息。只需要实例化该类，就可以访问类中的成员，从而获取类的基本信息。下面创建一个实例，演示如何获得类的基本信息。具体如代码 4.10 所示。

代码 4.10　使用 ReflectionClass 类

```
<?php
class MyClass
{   const definevalue = 5;
    private static $value = TestClass::definevalue;
       public function myTest() {  return self::$value++;  }
       public function myName(){echo "Yound Tang";}
}
$class = new ReflectionClass('MyClass');
$methods=$class->getMethods();
echo "MyClass 类的方法有: ";
foreach($methods as $method)
  echo $method->getName()."--";
$isAbstract=$class->isAbstract()?"是一个抽象类":"不是一个抽象类";
$isFinal=$class->isFinal()?"是一个最终类":"不是一个最终类";
echo("<br/>");
echo $class->getName()."类".$isAbstract."<br/>";
echo $class->getName()."类".$isFinal;
?>
```

保存并执行代码 4.10，页面效果如图 4-7 所示。

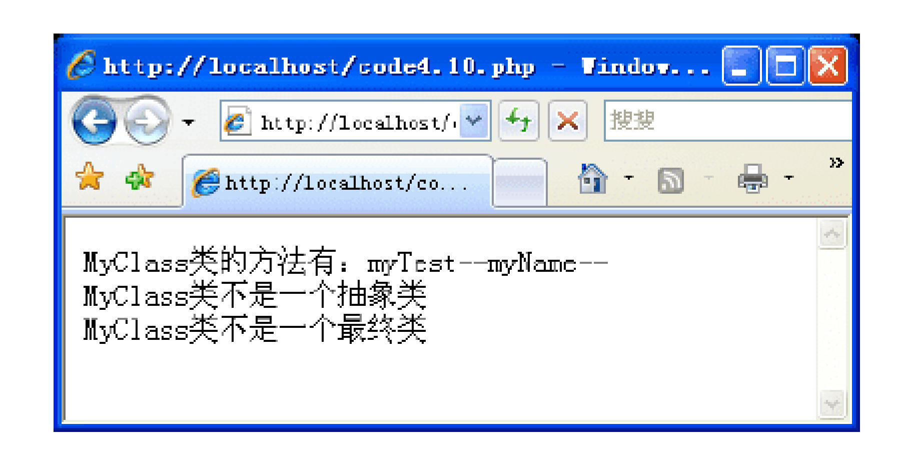


图 4-7　类反射

在本实例的代码中，创建了一个类 MyClass，在类里面创建了一个常量，一个静态变量，两个普通方法。使用语句$class =new ReflectionClass(' MyClass ')创建了反射类的实例化对象$class，然后使用 getMethods()方法获取类中的方法，接下来使用 getName()方法获取方法的名称，然后使用 isAbstract()方法判断该类是否为抽象类，最后使用 isFinal()方法判断该类是否为最终类。

4.6.2　编写 ReflectionMethod 类

ReflectionMethod 类主要用于获取方法的详细信息，例如该方法是否为抽象方法，是否为最终方法，是否静态方法，其作用域是什么等。同样，使用该类中的方法也需要实例化 ReflectionMethod 类，然后使用该类的对象访问类成员。下面创建一个实例，演示获取一个方法的详细信息。具体如代码 4.11 所示。

代码 4.11　获取一个方法的详细信息

```
<?php
class MyClass
{   const myAge = 5;
    private static $value = MyClass::myAge;
       public function GetAge() {  return self::$value++;  }
}
$method = new ReflectionMethod('MyClass','GetAge');
printf( "这个 %s 方法的特点: <br/>%s 方法<br/> %s 方法<br/>%s 方法<br/>".
     "'%s' (这个方法 %s)<br/>" .
    "    声明在:  %s<br/>" .
    "    行数从 : %d to %d<br/>" .
    "    具有的修饰符为:  %d[%s]<br/>",
        $method->isInternal() ? '系统内置的' : '用户自定义的',
        $method->isAbstract() ? ' 抽象' : '普通',
        $method->isFinal() ? ' 最终' : '不是最终',
        $method->isPublic() ? ' 公共' : '',
        $method->getName(),
        $method->isConstructor() ? '构造函数' : '普通方法',
        $method->getFileName(),
        $method->getStartLine(),
        $method->getEndline(),
        $method->getModifiers(),
        implode(' ', Reflection::getModifierNames($method->getModifiers())));
?>
```

保存并执行代码 4.11，页面效果如图 4-8 所示。

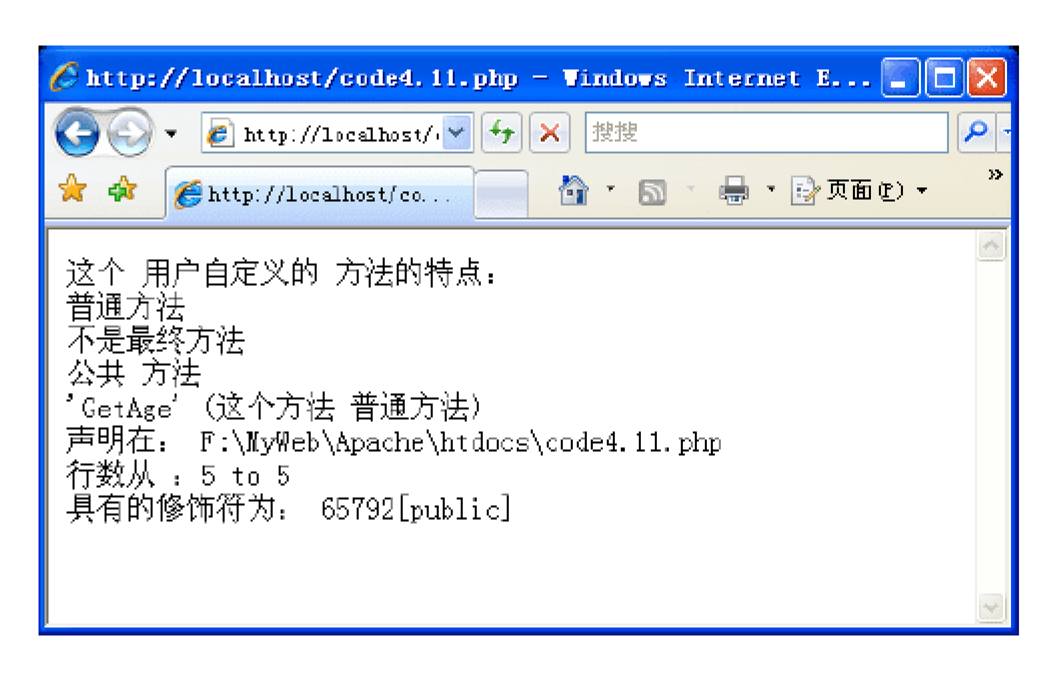


图 4-8　方法反射

4.6.3　编写 ReflectionParameter 类

ReflectionParameter 类获取一个函数或方法的参数信息，如参数的名称。要访问该类中的成员，首先需要实例化 ReflectionParameter 类，然后再使用对象访问类成员。下面创建使用

ReflectionParameter 类的实例，具体如代码 4.12 所示。

代码 4.12　使用 ReflectionParameter 类

```
<?php
function Yound($chinese, $math, $english) {}
function Joy($chinese, $math, $english) {}
function ruirui(ReflectionFunction $chinese, $math = 95, $english = null) {}
function Studentresult() {}
$reflect = new ReflectionFunction(Yound);
echo $reflect;
foreach ($reflect->getParameters() as $i => $param)
{
        printf( "-- Parameter #%d: %s {<br/>".
        "   Class: %s<br/>".
        "   Allows NULL: %s<br/>".
        "   Passed to by reference: %s<br/>".
        "   Is optional?: %s<br/>".
        "}<br/>",
        $i,
        $param->getName(),
        var_export($param->getClass(), 1),
        var_export($param->allowsNull(), 1),
        var_export($param->isPassedByReference(), 1),
        $param->isOptional() ? 'yes' : 'no');
}
?>
```

保存并执行代码 4.12，页面效果如图 4-9 所示。

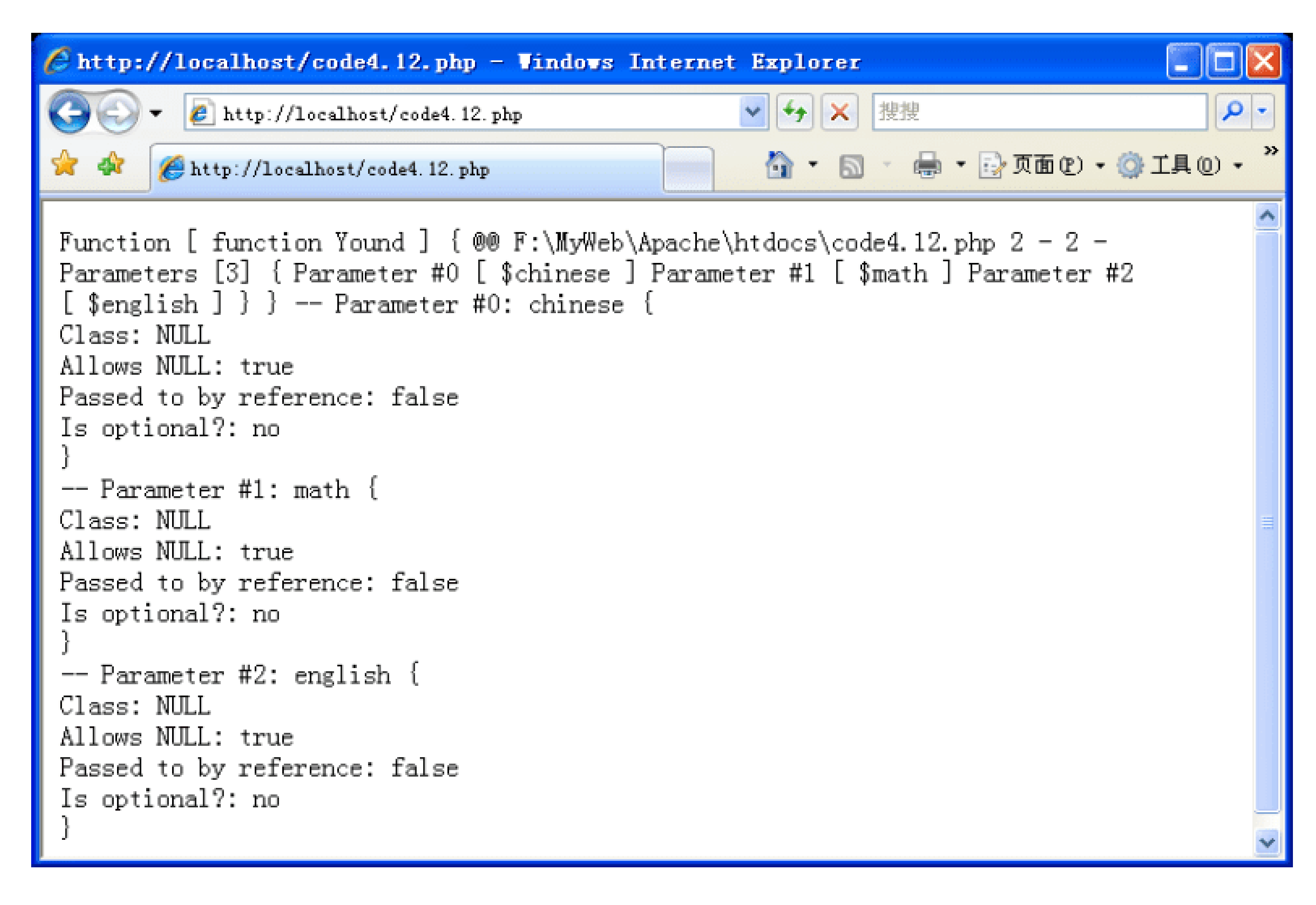


图 4-9　使用 ReflectionParameter 类

4.6.4　编写 ReflectionProperty 类

ReflectionProperty 类允许反向设计类属性，即获取类的属性的相关信息，例如获取属性的名称、数值和作用域范围等。首先创建 ReflectionProperty 类的一个实例，然后使用 ReflectionProperty 类方法才能获取这些信息，下面创建一个使用该类的示例，如代码 4.13 所示。

代码 4.13　使用 ReflectionProperty 类

```
<?php
class TestClass
{
    public  $myage =23 ;
}
$prop=new ReflectionProperty('TestClass','myage');
printf( "属性信息:  %s、%s、%s、%s <br/> 名称: '%s' (%s)<br/>" .
   " %s<br/>",
      $prop->isPublic() ? ' public' : '',
      $prop->isPrivate() ? ' private' : '',
      $prop->isProtected() ? ' protected' : '',
      $prop->isStatic() ? ' static' : '',
      $prop->getName(),
      $prop->isDefault() ? 'declared at compile-time' : 'created at run-time',
      var_export(Reflection::getModifierNames($prop->getModifiers()),1)
);
$obj= new TestClass();//创建一个 String 类的实例化对象
printf("它的值为: ");
var_dump($prop->getValue($obj));
$prop->setValue($obj,10);
printf("重新设置属性的值为 10: ");
var_dump($prop->getValue($obj));
var_dump($obj);
?>
```

保存并执行代码 4.13，页面效果如图 4-10 所示

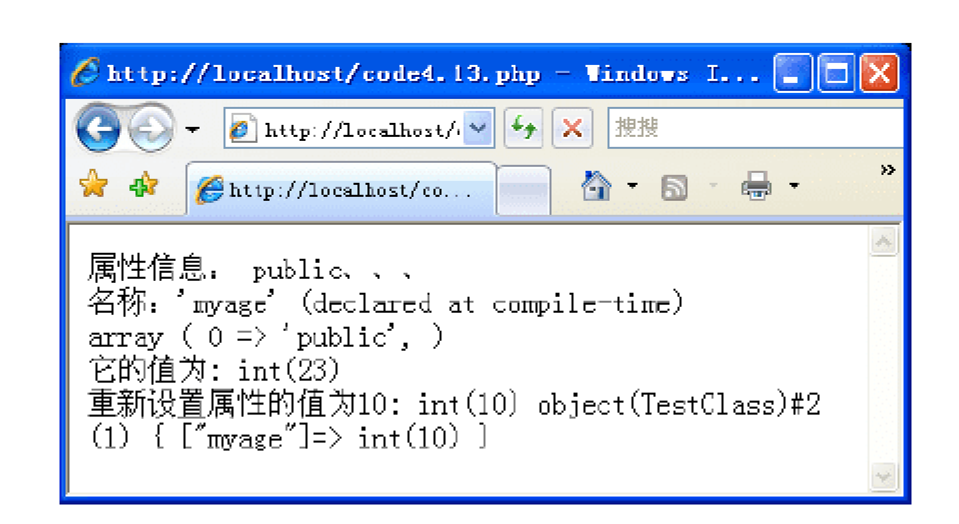


图 4-10　使用 ReflectionProperty 类

这里需要注意的是，试图获得或者设置私有或受保护类的属性值将抛出异常。

4.6.5 编写 ReflectionExtension 类

ReflectionExtension 类允许反向设计扩展。可以在使用 get_loaded_extensions()运行时，重新返回所有加载的扩展，即获得该类在运行时被加载的类的信息。ReflectionExtension 类被实例化后，通过对象可以访问其类成员。下面创建一个使用该类的示例，如代码 4.14 所示。

代码 4.14 使用 ReflectionExtension 类

```
<?php
$extension=new ReflectionExtension('date');
printf(
    "名称: %s<br/>" .
    "版本: %s<br/>" .
    "函数: [%d] %s<br/>" .
    "类名称: [%d] %s<br/>",
        $extension->getName(),
        $extension->getVersion()?$extension->getVersion():'NO_VERSION',
        sizeof($extension->getFunctions()),
        var_export($extension->getFunctions(),1),
        sizeof($extension->getClassNames()),
        var export($extension->getClassNames(),1)
);
?>
```

保存并执行代码 4.14，效果如图 4-11 所示。

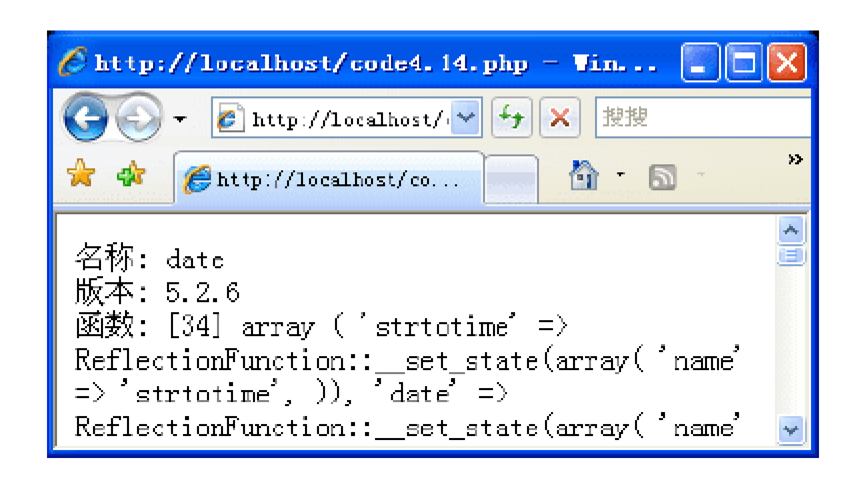
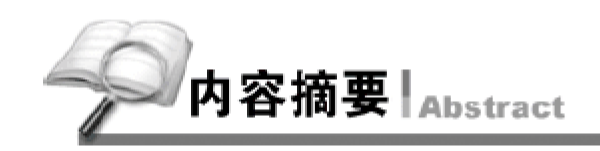


图 4-11 使用 ReflectionExtension 类

第 2 篇　PHP Web 应用篇

第5章 PHP 简单 Web 操作



数据交互是动态网站的一大特点，如果客户端不能与服务器端交互数据，那么就不能称为是动态网站。使用 PHP 创建的网站同样可以使客户端和服务器端交互数据，一个典型的应用就是用户向服务器端提交表单。用户提交表单后，PHP 会在服务器端处理用户提交的数据，还可以对提交信息进行验证和字符串处理。本章主要介绍 PHP 如何处理表单、处理字符串和 PHP 身份认证。



- 灵活运用 GET 和 POST 获取表单数据
- 掌握如何向函数传递表单数据
- 熟练处理多值表单组件
- 了解字符串相关函数
- 了解基本的 HTTP 身份验证
- 掌握 PHP 身份认证

5.1 PHP 和 Web 表单

在应用程序中，使用表单向应用程序输入原始数据，然后再提交表单，从而取得与服务器端的数据交互。本节主要介绍如何使用 Web 表单结合 PHP 获取和处理有价值的用户数据。

5.1.1 HTML 表单 GET 和 POST

当一个脚本向另一个脚本传递数据时，GET 方法是默认方法，但是一般希望使用 POST 方法，因为 POST 方法能处理更多的数据。本节主要介绍如何使用 POST 方法。

POST 方法发送给 PHP 脚本的提交数据都必须使用$_POST 语法来引用。假设表单中包含一个文本框 StudentName，使用 POST 方法提交数据时，先为该文本框赋值，然后使用$_POST 获取值。代码 5.1 为 POST 方法获取数据的示例。

代码 5.1　运用 POST 方法获取表单中数据

```
<html>
<head>
  <title>表单数据</title>
</head>
 <body marginheight="0">
 <form action="submitStudentName.php" method="POST">
    <input type="text" name="StudentName" size="20" maxlength="40" value="">
    <br/>
    <input type="submit" name="submit" value="在网页上显示数据">
 </form>
 </body>
</html>
<?php
if(isset($_POST['submit']))
{
   $name = $_POST['StudentName'];
   echo $name;
}
?>
```

代码 5.1 中加粗部分代码，表示使用$_POST 变量获取表单中的数据。$_POST 变量与$_GET 变量一样，采用这种方式获取外部变量。保存该段代码到 submitStudentName.php 文件，打开 IE 浏览器，在地址栏中输入http://localhost/submitStudentName.php，单击【转到】按钮打开该页面，在页面上添加相关信息，然后再单击【在网页上显示数据】按钮，页面效果如图 5-1 所示。

图 5-1　页面效果图

下面创建一个简单的示例，详细讲解如何使用 POST 方法获得表单数据。首先创建一个表单，并在表单中提示用户输入姓名、电子邮件、地址和联系方式，代码 5.2 为设计好的 HTML 表单。

代码 5.2　创建表单

```
<table width="900" align="center" border="1" class="table4"
          cellpadding="0" cellspacing="0">
     <tr><td><img src="title.jpg" alt="标题"></td></tr>
     <tr><td><table width="550" class="table4">
        <tr><td align="right">用户姓名: </td>
       <td><input name="userName" type="text" size="20"
            maxlength="40" value=""></td>
       <td align="right"> 电子邮件: </td>
       <td><input name="email" type="text" size="20"
            maxlength="40" value=""></td></tr>
     <tr>
```

```
        <td align="right">联系地址: </td>
        <td><input name="address" type="text" size="20"
            maxlength="40" value=""></td>
        <td align="right"> 联系电话: </td>
        <td><input name="phone" type="text" size="20"
            maxlength="40" value=""></td></tr>
    <tr>
    <td></td>
     <td colspan="3" align="center"><input type="submit" name="submit"
        value="提交数据"></td></tr></table></td></tr>
    </table>
```

用户根据提示填写完信息并提交后，将输入信息显示在浏览器窗口中。下面编写显示用户输入信息的 PHP 脚本应用程序，如代码 5.3 所示。

代码 5.3　使用 POST 方法获得表单数据

```
<?php
  if(isset($_POST['submit']))
  {
     echo "<li></li>你好~~!       ".$_POST['userName']."<br/>";
     echo "<li></li>你的电子邮件是: ".$_POST['email']."<br/>";
     echo "<li></li>你的联系地址是: ".$_POST['address']."<br/>";
     echo "<li></li>你的联系电话是: ".$_POST['phone']."<br/>";
  }
?>
```

在代码 5.3 中，首先使用$_POST 获得用户输入信息，然后再输出这些用户信息。保存代码，打开 IE 浏览器，在地址栏中输入http://localhost/UsePost.php，然后单击【转到】按钮，页面显示了一个让用户输入信息的表单，效果如图 5-2 所示。然后再根据提示输入相应的信息，最后单击【提交数据】按钮，效果如图 5-3 所示。



图 5-2　初始页面效果图



图 5-3　提交数据后页面效果图

5.1.2　向函数传递表单数据

在处理 PHP 应用程序时，往往需要向函数中传递用户输入的数据。向函数中传递表单数据与传递其他变量的过程一样，只需要将提交的表单数据作为函数参数传递即可。假设需要向前面的示例中加入一个服务器验证，可以使用一个定制的函数来确认电子邮件地址是否合法。下面修改 PHP 代码片段，并在其中添加一个函数用于验证电子邮件的合法性，如代码 5.4 所示。

代码 5.4　验证电子邮件合法性并处理提交数据

```
<?php
  function Authentication($email)
  {
     $regexp = "^([_a-z0-9-]+)(\.[_a-z0-9]+)*@([a-z0-9]+)(\.[a-z0-9-]+)*(\
    .[a-z]{2,6})$";
     if(eregi($regexp,$email))
     {
         return true;
     }
     else
     {
         return false;
     }
  }
  if(isset($_POST['submit']))
  {
    if(Authentication($_POST['email']))
    {
```

```
            echo "<li></li>你好~~!      ".$_POST['userName']."<br/>";
            echo "<li></li>你的电子邮件是: ".$_POST['email']."<br/>";
            echo "<li></li>你的联系地址是: ".$_POST['address']."<br/>";
            echo "<li></li>你的联系电话是: ".$_POST['phone']."<br/>";
        }
        else
        {
            echo"<li></li>该电子邮件: <strong>".$_POST['email']."</strong>不合法,
            请重新输入";
        }
    }
?>
```

在代码 5.4 中，首先创建了一个验证电子邮件的函数，并向该函数中传递输入电子邮件地址的参数。保存该文件，在 IE 浏览器地址栏中输入http://localhost/UsePost.php，单击【转到】按钮，在页面上输入错误的电子邮件地址，再单击【提交数据】按钮，页面效果如图 5-4 所示。



图 5-4　输入错误电子邮件地址页面效果图

5.1.3　处理多值表单组件

多值表单就是平常在页面上见到的复选框和下拉列表等，这些组件都允许用户为一个指定的表单项同时选择多个值，大大提高了 Web 的数据收集能力。本节将主要讲解如何使用 PHP 获取用户在复选框和下拉列表上选择的值。

下面创建一个示例，具体讲解如何使用 POST 获取多值表单组件的值，并显示到页面上。该示例为一个用户选择爱好页面，当用户选择完爱好后，进行提交，页面上会显示用户选择的爱好信息，代码 5.5 所示为该页面设计源程序。

代码 5.5　页面设计脚本

```
<html><head><title>选择爱好页面</title>
<link href="style.css" type="text/css" rel="stylesheet" />
</head><body>
<form action="Demo-Select.php" method="post">
```

```
<table border="0" cellpadding="0" cellspacing="0" align="center">
    <tr><td><img src="title.jpg" /></td></tr>
     <tr><td background="3.gif" height="23" style="font-size:16px; color:#FF
     FFFF">
       <strong> 请选择你喜欢的运动: </strong></td></tr>
     <tr><td>
     <fieldset><legend>爱好: </legend>
      <input type="checkbox" name="sports[]" value="篮球" />篮球
      <input type="checkbox" name="sports[]" value="足球" />足球
      <input type="checkbox" name="sports[]" value="排球" />排球
      <input type="checkbox" name="sports[]" value="台球" />台球
      <input type="checkbox" name="sports[]" value="标枪" />标枪<br/>
      <input type="checkbox" name="sports[]" value="跑步" />跑步
      <input type="checkbox" name="sports[]" value="太极拳" />太极拳
      <input type="checkbox" name="sports[]" value="举重" />举重
      <input type="checkbox" name="sports[]" value="游泳" />游泳
      <input type="checkbox" name="sports[]" value="飙车" />飙车
     </fieldset></td></tr>
     <tr>
       <td background="3.gif"  height="23" style="font-size:16px; color:#FF
       FFFF">
       <strong> 请选择你喜欢的计算机语言: </strong></td></tr>
     <tr><td><fieldset><legend>语言: </legend>
     <table cellpadding="0" cellspacing="0" border="0" width="100%">
     <tr><td width="15%">
     <select name="languages[]" multiple="multiple" style="width:50">
       <option value="C#">C#</option>
       <option value="Java">Java</option>
       <option value="PHP">PHP</option>
       <option value="C++">C++</option>
       <option value="VB">VB</option>
     </select></td><td>
     <?php
        if(isset($_POST['submit']))
       {
           echo "<strong>你爱好的运动是: </strong><br/>";
           foreach ($_POST['sports'] as $sport)
          {
           echo $sport."    ^!^    ";
          }
          echo "<br/>";
          echo "<strong>你喜欢的计算机语言是: </strong><br/>";
          foreach ($_POST['languages'] as $language)
```

```
            {
             echo "$language   ^!^   ";
            }
        }
      ?>
      </td></tr></table>
      </fieldset></td></tr>
      <tr> <td>
        <table border="0" width="100%" cellpadding="0" cellspacing="0">
        <tr><td background="3.gif" style="width:100px"></td>
         <td background="3.gif" > <input type="submit" name="submit" value="
          提交数据" /></td>
        </tr></table></td></tr>
   </table>
   </form></body></html>
```

在代码 5.5 中，首先创建了一个组复选框，在 PHP 中为了识别赋给一个表单变量的多个值，需要将表单多值组件的 name 属性命名为带有“[]”的名称。例如代码中为复选框命名为“sports[]”。如此命名之后，PHP 将像处理所有其他数组一样对待所提交的变量。保存该文件，打开浏览器，在地址栏中输入 http://localhost/Demo-Select.php，然后单击【转到】按钮，将展示选择爱好页面。在该页面上选择爱好，并单击【提交数据】按钮，页面效果如图 5-5 所示。



图 5-5　选择爱好之后页面效果图

5.2　PHP 与字符串

PHP 中有许多内置函数用于处理字符串，例如获取字符串的长度、字符串的比较等。这类函数有 100 多个，用来处理字符串的各个方面，本节主要介绍几种常用的处理字符串的函数。

5.2.1 获取字符串长度

获取一个字符串的长度，在 PHP 程序中应用得非常普遍。例如判断客户端输入密码的长度和获取客户端提交的留言信息长度等。获取字符串长度的函数是 strlen()，其语法格式为：

```
int strlen ( string $string )
```

该函数返回值为整型，表示字符串$string 的长度，如果返回值为 0 表示该字符串为空。下面创建一个示例，具体讲解如何使用 strlen()函数，如代码 5.6 所示。

代码 5.6　使用 strlen()函数

```
<?php
 function GetLength($str)
 {
    if(strlen($str) > 10)
    {
        echo "你要测试的字符串长度大于 10，值为: ".strlen($str);
    }
 }
 GetLength("I LOVE PHP~!");
?>
```

在上述代码中，使用 strlen()函数获取字符串的长度，如果一个字符串中包含空格，空格同样也被计算在内，保存该段代码并执行，页面显示结果如下所示：

```
你要测试的字符串长度大于 10，值为: 12
```

5.2.2 字符串比较函数

在任何语言中，比较两个字符串都是常用的功能之一，例如 C#语言。同样在 PHP 中也有处理两个字符串比较的函数，主要有 4 个，分别是 strcmp()函数、strcasecmp()函数、strspn()函数和 strcspn()函数。本节主要讲解的就是这 4 个函数，下面分别介绍。

1. strcmp()函数

该函数对两个字符串进行大小比较，并区分大小写，语法格式如下所示：

```
int strcmp(string str1,string str2)
```

由上述代码知，该函数返回 int 类型数据，如果 str1 和 str2 相等则返回 0，如果 str1 比 str2 大，则返回 1，如果 str1 比 str2 小，则返回–1。下面创建一个示例，具体讲解如何使用 strcmp()函数，以及在何时使用 strcmp()函数。

在网上注册账号时，往往要用户输入两遍密码，输入两遍的目的是减少由于键入错误而

生成不正确密码的可能性。因为密码通常区分大小写，所以使用 strcmp()函数比较两个密码是否相同非常合适。代码 5.7 所示为使用 strcmp()函数比较两次密码输入是否相同的示例。

代码 5.7　使用 strcmp()函数

```
<html>
  <head><title>运用 strcmp()函数</title>  </head>
  <body>
  <form action="Demo-strcmp.php" method="POST">
  <fieldset><table width="90%" cellpadding="0" cellspacing="0" border="0">
     <tr><td align="right">用户名: </td>
       <td><input type="text" name="userName" value=""></td>
     </tr>
     <tr><td align="right">密码: </td>
       <td><input type="password" name="userPwd" value=""></td>
     </tr>
     <tr><td align="right"> 重复密码: </td>
       <td><input type="password" name="twopwd" value=""></td>
     </tr>
     <tr><td></td>
       <td><input type="submit" name="submit" value="注册"></td>
     </tr>
     <tr><td colspan="2">
<?php
   if(isset($_POST['submit']))
   {
      if(strcmp($_POST['userPwd'],$_POST['twopwd'])==0)
      {
         echo "<hr/><li>恭喜你注册成功</li><br/>";
         echo "<li>你的用户名是: </li>".$_POST['userName'];
         echo "<br/><li>你注册的密码是: </li>".$_POST['userPwd'];
      }
      else
      {
         echo "<hr/><li></li>你注册的密码和重复密码不同，请重新输入你的密码";
      }
   }
?>
       </td></tr></table>
   </fieldset></form></body></html>
```

保存代码 5.7，并为该文件命名为“Demo-strcmp.php”，然后打开 IE 浏览器，在浏览器的地址栏中输入 http://localhost/Demo-strcmp.php，然后单击【转到】按钮。在该页面上，根据提示输入相关信息，当输入的密码和重复密码相同时，页面效果如图 5-6 所示，当输入的密码和

重复密码不同时，页面效果如图 5-7 所示。

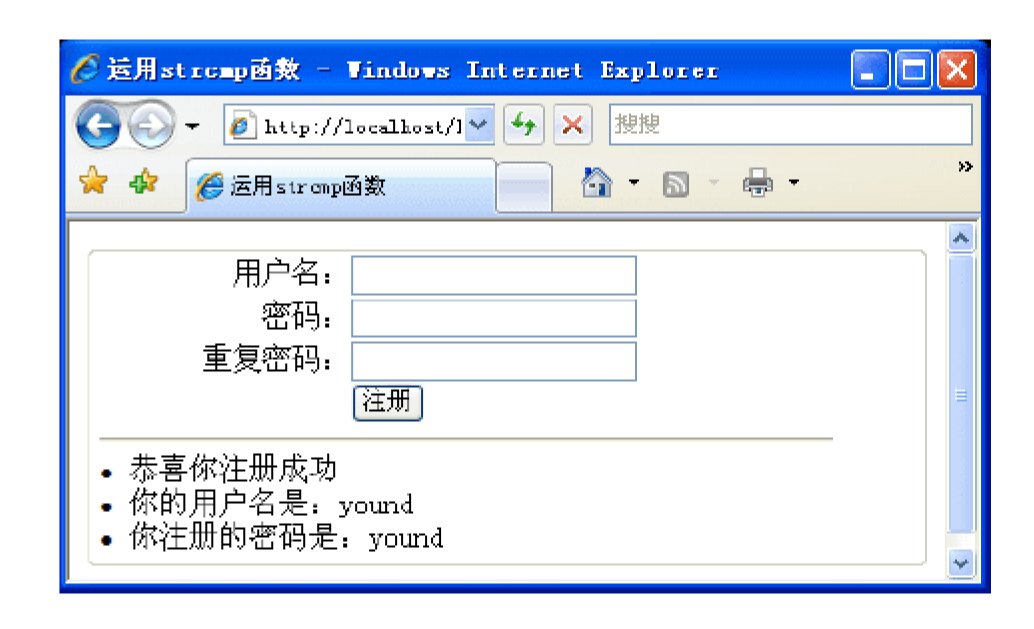




图 5-6　输入密码相同时页面效果图

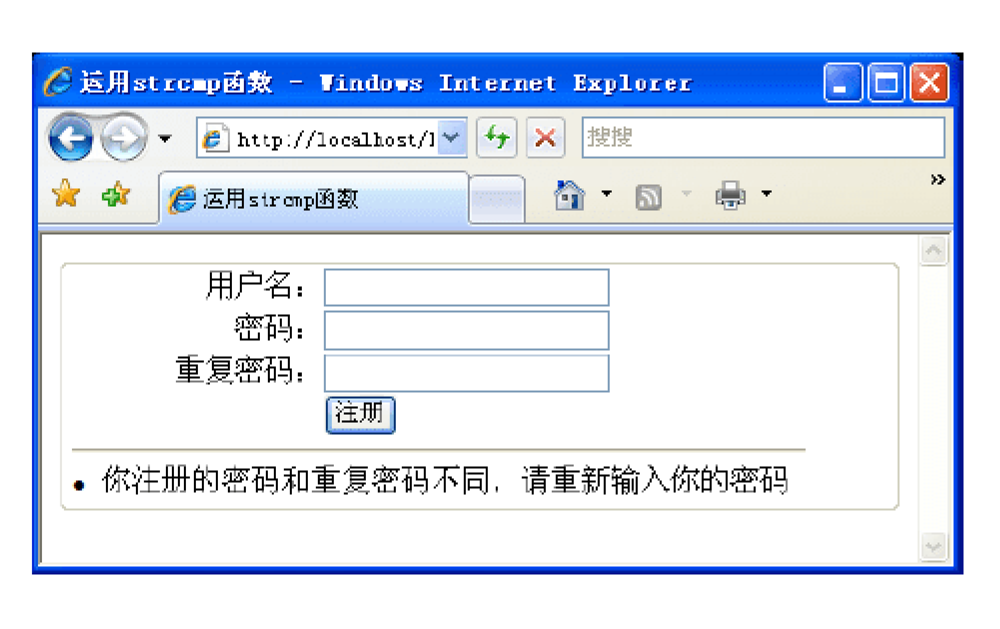


图 5-7　输入密码不同时页面效果图

2．strcasecmp()函数

该函数的功能与 strcmp()基本相同，只是 strcasecmp()在比较字符串时不区分大小写，语法格式如下所示：

```
int strcasecmp(string str1,string str2)
```

下面创建一个简单的示例，具体说明如何运用 strcasecmp()函数，如代码 5.8 所示。

代码 5.8　运用 strcasecmp()函数

```
<?php
   $str1 ="yound";
   $str2 = "Yound";
   if(strcasecmp($str1,$str2)==0)
   {
      echo "这两个字符串相同，strcasecmp()函数比较时不区分大小写";
   }
   else
   {
      echo "这两个字符串不相同，strcasecmp()函数区分大小写";
   }
?>
```

保存代码 5.8，打开浏览器执行该文件，结果如下所示：

```
这两个字符串相同，strcasecmp()函数比较时不区分大小写
```

3．strspn()函数

strspn()函数返回第一个参数中匹配第二个参数中的单字符串连续出现长度的值，该函数的第三个参数和第四个参数可选，分别表示开始匹配的位置和返回的最大值，该函数的语法格式如下所示：

```
int strspn(string str1,string str2,int start,int length)
```

下面创建一个示例，具体说明该函数的运用，如代码 5.9 所示。

代码 5.9　运用 strspn()函数

```
<?php
 $length = strspn("123567890","1234567980");
 echo $length."<br/>";                        //返回值为 9
 echo strspn("admin" ,"a")."<br/>";           //显示值为 1
 echo strspn("yound","you")."<br/>";          //显示值为 3
 echo strspn("yound","u");                    //显示值为 0
?>
```

4．strcspn()函数

该函数返回第一个参数中包含第二个参数中所没有字符的第一部分的长度，语法格式如下所示：

```
int strcspn(string str1,string str2)
```

下面创建一个示例，具体说明如何运用 strcspn()函数，如代码 5.10 所示。

代码 5.10　使用 strcspn()函数

```
<?php
  $password = "a1234";
  if(strcspn($password,"1234567890")!=0)
  {
     echo "该密码中包含了非数字字符";
  }
?>
```

5.2.3　字符串大小写转换

在其他语言中，都有方法或者函数处理字符串的大小写转换，同样在 PHP 中也有功能相同的函数，分别是：strtolower()函数、strtoupper()函数、ucfirst()函数和 ucwords()函数，表 5-1 列出了这 4 个函数的功能和语法格式。

表 5-1　大小写转换函数

名称	语法格式	功能
strtolower()	string strtolower(string $str)	将字符串的字母全部转换为小写字母
strtoupper()	string strtoupper(string $str)	将字符串的字母全部转换为大写字母
ucfirst()	string ucfirst(string $str)	如果字符串的第一个字符是字母，该函数则将其变成大写，非字母字符不受影响
ucwords()	string ucwords(string $str)	该函数将字符串中每个单词的第一个字母变为大写，非字母字符不受影响

在表 5-1 中，详细介绍了处理字符串大小写转换的函数，以及函数的语法格式和功能。下面利用这些函数创建一个示例，具体讲解如何使用这些函数转换字符串大小写，如代码 5.11 所示。

代码 5.11　使用字母大小写转换函数

```
<html>
 <head><title>使用转换函数</title></head>
 <body>
  <form action="Demo-lowerorupper.php" method="POST">
  <fieldset><legend>转换字符串</legend> <table>
     <tr><td align="right">全部转换为小写的字符串：</td>
       <td><input type="text" name="lower" value="" /></td>
       <td>
         <?php
            if(isset($_POST['submit']))
            {
              echo strtolower($_POST['lower']);
            }
         ?>
       </td></tr>
     <tr><td align="right">全部转换为大写的字符串：</td>
       <td><input type="text" name="upper" value="" /></td>
       <td>
         <?php
            if(isset($_POST['submit']))
            {
              echo strtoupper($_POST['upper']);
            }
         ?>
       </td></tr>
     <tr><td align="right">字符串首字母转换为大写的字符串：</td>
       <td><input type="text" name="first" value="" /></td>
       <td>
         <?php
            if(isset($_POST['submit']))
            {
              echo ucfirst($_POST['first']);
            }
         ?>
       </td></tr>
     <tr><td align="right">字符串中每个单词首字母转换为大写的字符串：</td>
       <td><input type="text" name="words" value="" /></td>
       <td>
         <?php
```

```
            if(isset($_POST['submit']))
            {
              echo ucwords($_POST['words']);
            }
        ?>
      </td></tr>
    <tr>
      <td colspan="3" align="center">
            <input type="submit" name="submit" value="转换" />
      </td></tr></table></fieldset></form>
      </body></html>
```

保存代码 5.11，并重命名该文件为 Demo-lowerorupper.php。打开 IE 浏览器，在地址栏中输入 http://localhost/Demo-lowerorupper.php，单击【转到】按钮，在页面上根据提示输入相关信息，页面效果如图 5-8 所示，然后再单击【转换】按钮，页面效果如图 5-9 所示。



图 5-8　转换前页面效果图

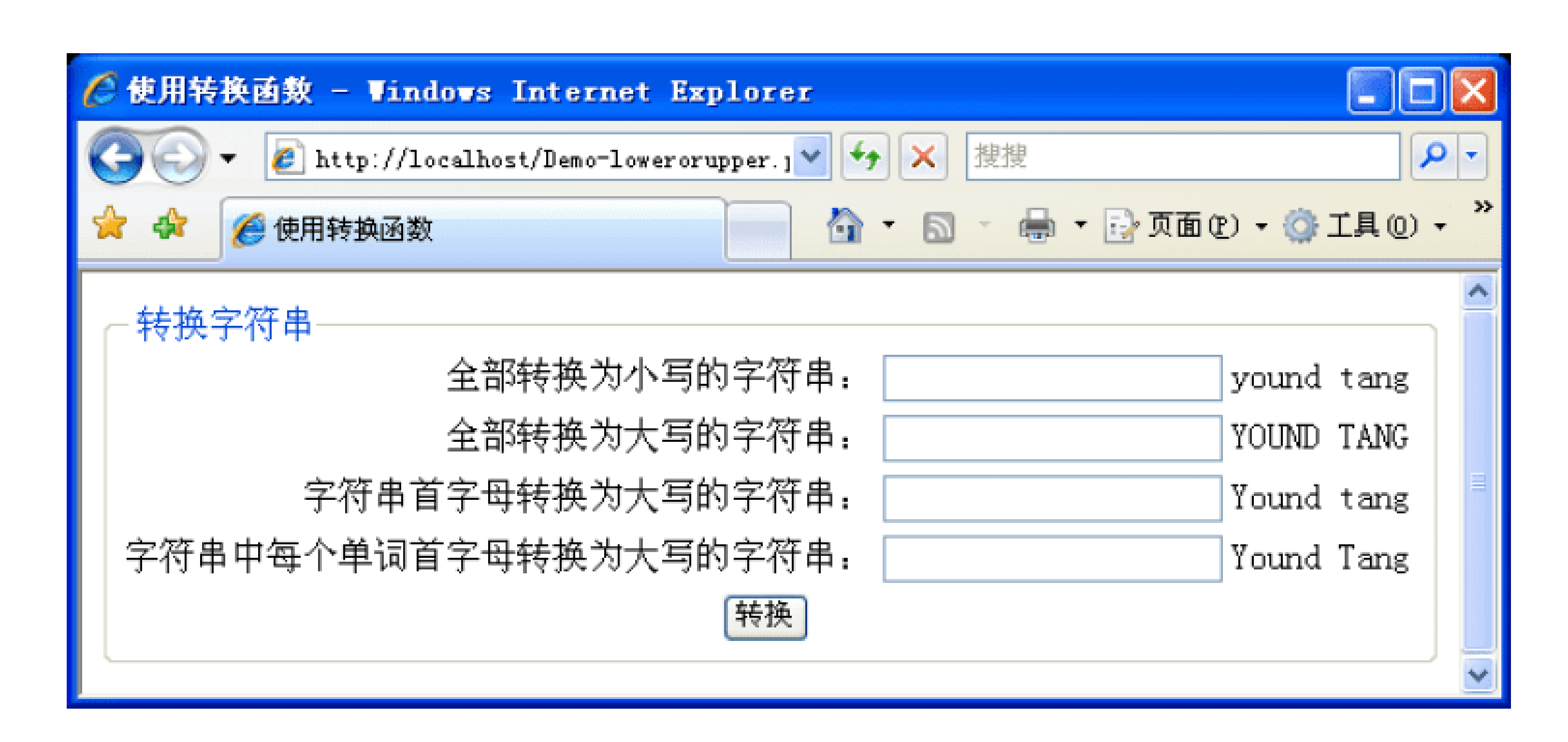


图 5-9　转换后页面效果图

5.2.4　填充和剔除字符串

有时，为了 Web 程序的需要，需要将字符串中存在的一些字符从字符串中去掉，或者在

字符串中添加一些新的字符。为实现该功能，PHP 提供了一些函数，本节将一一介绍。

1. ltrim()函数

ltrim()函数去除字符串左侧的空白或者指定的字符，这些指定的字符包括空格符、水平制表符（\t）、换行符（\n）、回车符（\r）、空值（\0）以及垂直制表符（\x0b）。如果去掉其他的字符，如字符串中的 a 字符，可以使用第二个参数。该函数语法格式如下所示：


```
string ltrim ( string $str [, string $charlist] )
```

下面创建一个示例，具体讲解如何使用该函数，如代码 5.12 所示。

代码 5.12　使用 ltrim()函数

```
<?php
$text = "\t\tHello World~! ";
$hello  = "Hello World";
$trimmed = ltrim($text);
echo $trimmed;
echo "<hr>";
$trimmed1= ltrim($text, " \t.");
echo $trimmed1;
echo "<hr>";
$trimmed2 = ltrim($hello, "Hdle");
echo $trimmed2;
?>
```

保存代码 5.12，运行程序，结果如下所示：

```
Hello World~!
---
Hello World~!
---
o World
```

2. rtrim()函数

该函数和 ltrim()函数功能相同，只是它从字符串的右侧删除字符。该函数的使用示例如下所示：

```
<?php
$str = "You are welcome to use rtrim () function ";
echo $str."<br/>";
echo rtrim($str);
?>
```

执行该段代码，结果如下所示：

```
You are welcome to use rtrim () function
You are welcome to use rtrim () function
```

在执行结果中输出的两行信息并不相同，输出的第一行字符串最后拥有空格，而用 rtim()函数后输出的字符串却没有空格。

3．trim()函数

trim()函数是 ltrim()和 rtrim()函数的组合，该函数可以从字符串的两侧删除字符串中的指定字符。使用示例如下所示：

```
<?php
echo trim("what a beautiful day today!","! adlotwy");//显示 hat a beautifu
?>
```

4．str_pad()函数

str_pad()函数的作用是对字符串进行两侧的补白，该函数有 4 个参数。第一个参数表示被操作的字符串对象；第二个参数表示补白以后的长度；第三个参数可选，表示进行补白的字符串，默认为空格；第四个参数可选，表示补白的方式，有 3 个常量可以选择：STR_PAD_RIGHT，STR_PAD_LEFT 和 STR_PAD_BOTH，分别表示右补白，左补白和两侧补白，默认值为 STR_PAD_RIGHT。该函数的使用示例如代码 5.13 所示。

代码 5.13　使用 str_pad()函数

```
<?php
$input = "Yound";
echo str_pad($input, 10);
echo "<br/>";
echo str pad($input, 10, "-=", STR PAD LEFT);
echo "<br/>";
echo str_pad($input, 10, "_", STR_PAD_BOTH);
echo "<br/>";
echo str_pad($input, 6 , "___");
?>
```

保存代码 5.13，并打开浏览器，使页面转到该文件所在位置，显示结果如下所示：

```
Yound
-=-=-Yound
__Yound___
Yound_
```

5.2.5　字符和单词计数

在应用程序中，确定给定字符串中字符或单词的总数往往很有必要。PHP 在字符串解析方面提供了强大的功能，使得这些任务非常简单。本节将主要介绍字符和单词计数的两个函

数，下面分别介绍。

1. count_chars()函数

该函数主要用来统计 string 中每个字节值出现的次数，使用多种模式返回结果。可选参数 mode 默认值为 0，并且函数的行为也依赖于该参数。语法格式如下所示：

```
mixed count_chars ( string $string [, int $mode] )
```

当 count_chars()函数的参数 mode 值不同时，函数所表达的功能也不相同。下面详细介绍当 mode 参数取不同值时函数的功能。

- **0** 以每个字符值作为键名，出现次数作为值的数组。
- **1** 与 0 相同，但只列出出现次数大于零的字符值。
- **2** 与 0 相同，但只列出出现次数等于零的字符值。
- **3** 返回由所有使用了的字符值组成的字符串。
- **4** 返回由所有未使用的字符值组成的字符串。

下面创建一个示例，具体说明如何运用 count_chars()函数，如代码 5.14 所示。

代码 5.14 运用 count_chars()函数

```
<?php
$data = "The use of count_chars () function";
$chart = count_chars($data,1);
foreach ($chart as $s=>$num)
{
    echo "字符".chr($s)."出现".$num."<br/>";
}
?>
```

执行该段代码，结果如下所示：

```
字符 出现 5
字符(出现 1
字符)出现 1
字符 T 出现 1
字符_出现 1
字符 a 出现 1
字符 c 出现 3
字符 e 出现 2
字符 f 出现 2
字符 h 出现 2
字符 i 出现 1
字符 n 出现 3
字符 o 出现 3
字符 r 出现 1
```

```
字符 s 出现 2
字符 t 出现 2
字符 u 出现 3
```


2. str_word_count()函数

str_word_count()函数的作用是获取字符串里面的英文单词总数信息，该函数语法格式如下所示：

```
mixed str_word_count ( string $string [, int $format [, string $charlist]] )
```

由函数语法格式知，第一个参数表示要操作的字符串对象，第二个参数可选，如果定义了第二个参数，那么参数值不同，函数所表示的意义也不同，省略的默认值是 0，参数值含义介绍如下。

- 0 表示只返回单词的个数。
- 1 表示将找到的单词作为一个数组返回。
- 2 表示找到的单词和单词所在的字符位置信息作为一个联合数组（associative）返回。

第三个参数也是可选参数，表示需要忽略的断词符号，默认的断词符号有空格和数字等。下面创建一个示例，具体讲解如何使用该函数，如代码 5.15 所示。

代码 5.15　使用 str_word_count()函数

```
<?php
$summary ="We do not live in arrears anything,
      so there is no need to suffer gravely total.
      Life should be grateful,
      at least it gives us life,
      gave us survive.";
$word = str_word_count($summary);
echo "该语句共有".$word."个单词";
echo "<br/>";
$words = str_word_count($summary,2);
$frequency = array_count_values($words);
print_r($frequency);
?>
```

在代码 5.15 中，使用了 arry_count_values()函数，执行结果如下所示：

```
该语句共有 29 个单词
Array ( [We] => 1 [do] => 1 [not] => 1 [live] => 1 [in] => 1 [arrears] => 1
[anything] => 1 [so] => 1 [there] => 1 [is] => 1 [no] => 1 [need] => 1 [to]
=> 1 [suffer] => 1 [gravely] => 1 [total] => 1 [Life] => 1 [should] => 1 [be]
=> 1 [grateful] => 1 [at] => 1 [least] => 1 [it] => 1 [gives] => 1 [us] => 2
[life] => 1 [gave] => 1 [survive] => 1 )
```

5.2.6 字符串与 HTML 相互转换

在 PHP 中，可以使用一些函数将字符串或者整个文件转换为适合于 Web 形式的文本信息。PHP 中提供两类这样的函数：一类是将纯文本转换为 HTML，一类是将 HTML 转换为纯文本。本节主要介绍这些函数。

1. 将纯文本转换为 HTML

将纯文本转换为 HTML 的信息，需要使用 nl2br()函数、htmlentities()函数、htmlspecialchars()函数、get_html_translation_table()函数和 strtr()函数，下面以示例的方式详细讲解这些函数。

- **nl2br()函数**

nl2br()函数会将字符串中的换行符“\n”替换成“
”，并返回修改后的字符串。该函数的语法格式如下所示：

```
string nl2br ( string $string )
```

下面使用该函数创建一个示例，具体讲解如何使用 nl2br()函数，代码如下所示：

```
<?php
 $str= "This is a test function of the statement. \n Do you Know?";
 $s = nl2br($str);
 echo $s;
?>
```

执行该示例，查看页面源代码，结果如下所示：

```
This is a test function of the statement. <br/> Do you Know?
```

- **htmlentities()函数**

htmlentities()函数可以将字符串中一些字符转换为 HTML 实体，默认情况下主要包括这 4 个字符：“<”，“>”，“&”和“"”，该函数语法格式如下所示：

```
string htmlentities ( string $string [, int $quote_style [, string $charset]] )
```

htmlentities()函数的第二个可选参数表示可以选择引号的转换模式，可以选择 3 个常量：ENT_COMPAT 表示对双引号进行编码，不对单引号进行编码；ENT_QUOTES 表示同时对单引号和双引号进行编码；ENT_NOQUOTES 表示两个都不转换，默认值为 ENT_COMPAT。第三个可选参数表示所转换字符的编码集。其常用编码如表 5-2 所示。

表 5-2 字符编码集

字符集	别名	描述
ISO-8859-1	ISO8859-1	西欧，Latin-1
ISO-8859-15	ISO8859-15	西欧，Latin-9。增加了 Latin-1（ISO8859-1）中缺少的欧元符号、法国及芬兰字母
UTF-8		ASCII 兼容多字节 8-bit Unicode
cp866	ibm866, 866	DOS-特有的 Cyrillic 字母字符集。PHP 4.3.2 开始支持该字符集

续表

字符集	别名	描述
cp1251	Windows-1251,win-1251, 1251	Windows-特有的 Cyrillic 字母字符集。PHP 4.3.2 开始支持该字符集
cp1252	Windows-1252, 1252	Windows 对于西欧特有的字符集
KOI8-R	koi8-ru, koi8r	俄文。PHP 4.3.2 开始支持该字符集
BIG5	950	繁体中文，主要用于中国台湾
GB2312	936	简体中文，国际标准字符集
BIG5-HKSCS		繁体中文，BIG5 的延伸，主要用于香港
Shift_JIS	SJIS, 932	日文
EUC-JP	EUCJP	日文

下面创建一个示例，具体讲解如何使用该函数转换语句，如代码 5.16 所示。

代码 5.16　使用 htmlentities()函数

```
<?php
$str = "Hello <b>PHP</b>";
echo htmlentities($str);
echo "<br/>";
echo htmlentities($str, ENT_QUOTES);
?>
```

代码 5.16 执行后，可以看到在页面显示的形式与定义时的字符串相同，此时查看源代码，结果如下所示：

```
Hello &lt;b&gt;PHP&lt;/b&gt;<br/>
Hello &lt;b&gt;PHP&lt;/b&gt;
```

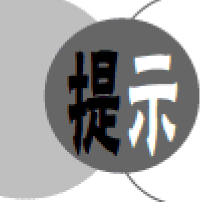
□ **htmlspecialchars()函数**

本函数将特殊字符转成 HTML 的字符串格式，语法格式如下所示：

```
string htmlspecialchars ( string $string [, int $quote_style [, string
$charset]] )
```

该函数最常用到的场合是处理客户留言的留言板。此函数只转换下面的特殊字符，分别是：“&”（和）转成“&amp”、“"”（双引号）转成“"”、“<”（小于）转成“<”、“>”（大于）转成“&gt”。对于该函数的第二个参数和第三个参数的含义，同 htmlentities()函数中的参数含义相同。该函数的使用示例如下所示：

```
<?php
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new; // &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;
?>
```



如果 htmlspeicalchars()要与 nl2br()等函数结合使用，应当在 htmlspeicalchars()之后再执行 nl2br()；否则，nl2br()生成的
标记将被转换为可见字符。

□ **get_html_translation_table()函数**

get_html_translation_table()函数是将文本转换为等价形式的一个遍历方法，该函数返回由table 指定的两种转换表之一，语法格式如下所示：

```
array get_html_translation_table ( [int $table [, int $quote_style]] )
```

由函数的语法格式知，get_html_translation_table()函数有两个常量参数。第一个参数表示选择显示哪种转换模式下的内容，该参数包括两种模式：HTML_ENTITIES 表示大范围的htmlentities()函数所用到的转换内容；HTML_SPECIALCHARS 表示小范围的 htmlspecialchars()函数所用到的转换内容。第二个参数具有 3 种模式 ENT_COMPA、ENT_QUOTES 和ENT_NOQUOTES。下面创建一个使用 get_html_translation_table()函数的示例，如代码 5.17 所示。

代码 5.17　使用 get_html_translation_table()函数

```
<?php
print_r(get_html_translation_table(HTML_SPECIALCHARS,ENT_QUOTES));
echo "<hr/>";
print_r(get_html_translation_table(HTML_ENTITIES,ENT_QUOTES));
?>
```

□ **strtr()函数**

该函数将第一个参数中的所有字符转换为第二个参数中相应的值，语法格式如下所示：

```
string strtr ( string $str, string $from, string $to )或
string strtr ( string $str, array $replace_pairs )
```

第一种形式该函数使用 3 个参数，替换时会将第二个参数的字符串替换为第三个参数的字符串（如果两个字符串参数的长度不一致，则较长的那个字符串将会被截断）。第二种方式使用两个参数，第二个参数是一个准备进行替换处理的数组。下面使用该函数创建一个示例，如代码 5.18 所示。

代码 5.18　使用 strtr()函数

```
<?php
$trans = strtr("hello world", "world", "body");
echo $trans;
echo "<br/>";
$arr = array("<b>" => "<strong>", "</b>" => "</strong>");
$trans = strtr("<b>This is a test function of the statement</b>", $arr);
echo $trans;
?>
```

执行代码 5.18，结果如下所示：

```
heyyo bodyd
This is a test function of the statement
```

在代码 5.18 的第 1 行代码中，第二个参数和第三个参数指定替换模板，这里会将字符“w”替换为“b”，字符“o”替换为“o”，字符“r”替换为“d”以此类推。模板获得后，就可以对第一个参数中的字符进行替换。

2．将 HTML 转换为纯文本

PHP 可以将 HMTL 标记的信息转换为纯文本信息，下面这个函数就可以完成这项功能。strips_tags()函数的主要功能是去掉 HTML 及 PHP 的标记。该函数的语法格式为：

```
string strip_tags ( string $str [, string $allowable_tags] )
```

strip_tags()函数的作用是去除一个字符串里面的 HTML 和 PHP 代码。第二个参数表示允许出现的标签对。如果字符串的 HTML 及 PHP 标签原来就有错，则返回错误。本函数和 fgetss()有着相同的功能，使用示例如下所示：

```
<?php
$text = '<p>Test paragraph.</p><em>Other text</em>';
echo strip_tags($text, '<p>'); // 显示 <p>Test paragraph.</p>Other text
?>
```

5.2.7　解析字符串表达式的函数

在 PHP 中可以对字符串进行匹配、替换和拆分等，这些功能都通过 PHP 中的一些函数实现，例如 strtok()函数、parse_str()函数、explode()函数和 str_replace()函数等。本节将详细讲解这些函数。

1．strtok()函数

strtok()函数的主要功能是指定若干个字符分割字符串，语法格式如下所示：

```
string strtok ( string $str, string $token )
```

strtok()函数将字符串 str 依字符串 token 的值切开分成小段的字符串。在使用该函数时，必须连续地调用才能完全地实现对一个串进行词法分析；每次调用该函数只是对字符串的下一部分做词法分析，但是 str 参数只需要指定一次，因为函数会跟踪 str 中的位置，直到完全完成了对 str 的词法分析，或者指定了新的 str 参数。下面创建一个示例，具体说明 strtok()函数的运用，如代码 5.19 所示。

代码 5.19　使用 strtok()函数

```
<?php
$str = "This is.a use of /strtok() /function test";
 $tokens = "./";
 $tokenized = strtok($str, $tokens);
while ($tokenized)
{
```

```
    echo "$tokenized<br />";
    $tokenized = strtok($tokens);
}
?>
```

在代码 5.19 中的分隔符分别是“.”和“/”，执行该段代码，结果如下所示：

```
This is
a use of
strtok()
function test
```

2. parse_str()函数

parse_str()函数的作用是将一定格式的字符串转变为变量和值，并将变量设置在当前作用域内，字符串的格式和 URL 的格式相同。该函数的语法格式如下所示：

```
void parse_str ( string $str [, array &$arr] )
```

在处理 URL 时，如果其中包含 HTML 表单或者其他通过查询字符串传递过来的参数，使用该函数特别有用。第二个参数是可选的，表示将转换后的值存到数组中。下面创建一个使用 parse_str()函数的示例，如代码 5.20 所示。

代码 5.20　使用 parse_str()函数

```
<?php
$str = "first=你好&arr[]=Yound&arr[]=Tang";
parse_str($str);
echo $first."<br/>";
echo $arr[0]." ";
echo $arr[1]."<br/>";
parse_str($str, $output);
echo $output['first']."<br/>";
echo $output['arr'][0]." ";
echo $output['arr'][1]."<br/>";
?>
```

这里需要注意的是，字符串的开始位置如果是字符“？”，将会无法解析出该字符串的第一个变量和值。执行该段代码，结果如下所示：

```
你好
Yound Tang
你好
Yound Tang
```

3. explode()函数

explode()函数使用一个字符串分割另一个字符串，并且该函数返回字符串组成的数组，每

个元素都是 str 的一个字符串，该函数的语法格式如下所示：

```
array explode ( string $separator, string $str [, int $limit] )
```

该函数将字符串 str 分成字符串数组，原字符串根据 separator 指定的字符分隔符分割为不同的元素，如果设置了 limit 参数，则返回的数组最多包含 limit 个元素，而最后那个元素将包含 str 的剩余部分。如果 separator 为空字符串（""），explode()将返回 FALSE。如果 separator 所包含的值在 str 中找不到，那么 explode()将返回包含 str 单个元素的数组。如果 limit()参数是负数，则返回除了最后的-limit 个元素外的所有元素。还可以结合 explode()函数、sizeof()函数和 strip_tags()函数共同确定文本块中单词的总数，下面创建一个使用 explode()函数的示例，该示例如代码 5.21 所示。

代码 5.21　使用 explode()函数

```
<?php
 $str = "Today, we are feeling pretty good, hope that we will be happy to learn
 PHP";
 $string = explode(" ",$str);
 foreach ($string as $s)
 {
    echo $s."<br/>";
 }
 $num = sizeof(explode(' ',$str));
 echo "该句话共有".$num."个字符串";
?>
```

在代码 5.21 中，首先创建一个字符串，接下来使用函数 explode()用空格字符串分割字符串，然后使用 echo 函数打印到页面上，最后使用 sizeof()函数获取字符串中单词个数。执行代码，页面效果如图 5-10 所示。

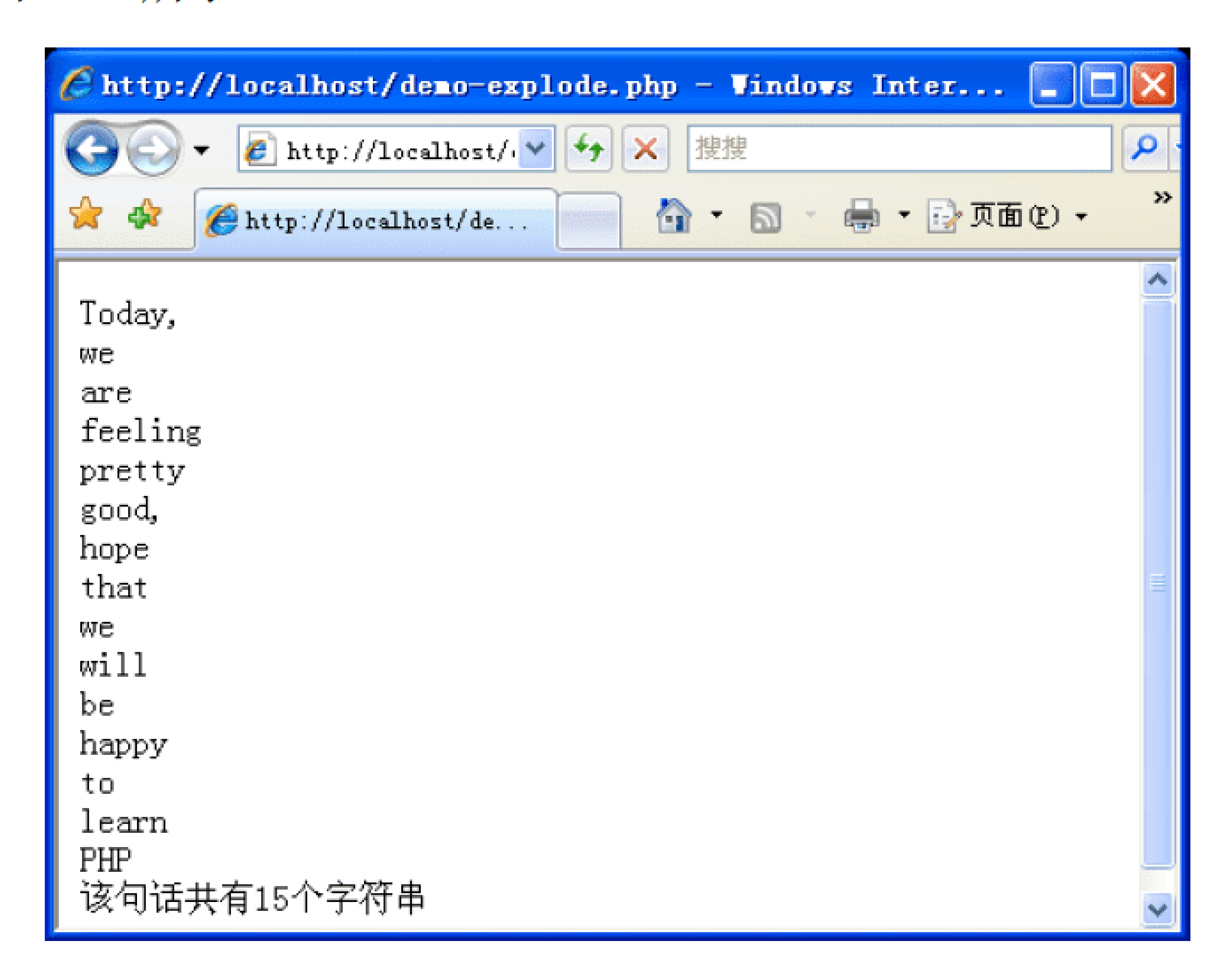


图 5-10　页面执行效果图

4．implode()函数

implode 函数将数组的内容组合成一个字符串，也可以说是使用定界字符串将数组元素连接成一个定界字符串，该函数语法格式如下所示：

```
string implode ( string $glue, array $pieces )
```

由上语法格式知 glue 为定界字符串，该函数使用 glue 定界字符串可以将数组组成一个字符串，上面创建一个示例，具体讲解如何使用 implode()函数，如代码 5.22 所示。

代码 5.22　使用 implode()函数

```
<?php
$array = array("My","Name","is","Yound Tang");
 $FirstImplode = implode("/",$array);
 echo $FirstImplode;
?>
```

在代码 5.22 中，首先定义了一个数组，使用 implode()函数将数组组合成字符串，该段代码执行结果如下所示：

```
My/Name/is/Yound Tang
```

5．strpos()函数

strpos()函数查找并返回首个匹配项的位置，语法格式如下所示：

```
int strpos ( string $haystack, mixed $needle [, int $offset] )
```

该函数在字符串 haystack 中以区分大小写的方式找到 needle 的第一次出现。haystack 参数表示处理的字符串；needle 参数表示匹配项；offset 表示开始执行查找的位置，并且 offset 是可选的参数。下面使用该函数创建一个示例，如代码 5.23 所示。

代码 5.23　使用 strpos()函数

```
<?php
$str = "Today is a great day";
 $num = strpos($str,"a");
 echo $num."<br/>";
 $num1 = strpos($str,"a",4);
 echo $num1;
?>
```

在代码 5.23 中，首先创建一个字符串，然后使用函数 strpos()查找字符 a 的位置，执行该段代码，结果如下所示：

```
3
```

```
9
```

6．stripos()函数

该函数的功能和 strpos 函数类似，只是该函数不区分匹配字符的大小写，该函数的语法格式如下所示：

```
int stripos ( string $haystack, string $needle [, int $offset] )
```

该函数和 strpos()函数的用法基本相同，参数所表示的意义也相同，在此就不再提供该函数的使用示例。

7．strrpos()函数

该函数跟前面两个函数的功能刚好相反，是查找字符在字符串中最后出现的位置，语法格式如下所示：

```
int strrpos ( string $haystack, mixed $needle [, int $offset] )
```

该函数的参数和前面两个函数的参数所表达的意思相同，在此就不再赘述。下面创建一个示例，具体讲解如何使用该函数，如下所示。

```
$str = "Today is a great day";
 $num = strrpos($str,"a");
 echo $num."<br/>";
 $num1 = strrpos($str,"a",5);
 echo $num1;
```

执行该段代码，结果如下所示：

```
18
18
```

8．str_replace()函数和 str_ireplace()函数

str_replace()函数查找并替换字符串中匹配的字符串，str_ireplace()函数和 str_replace()的功能相同，只是该函数能够执行不区分大小写的搜索，这两个函数的语法格式如下所示：

```
mixed str_replace (mixed $search, mixed $replace, mixed $subject [, int &$count] )
mixed str_ireplace ( mixed $search, mixed $replace, mixed $subject [, int
&$count] )
```

由函数语法格式知道，这两个函数可以替换固定的字符，str_replace()函数在 subject 中以区分大小写的方式搜索 search，然后使用 replace 替换找到的所有实例，如果没有找到 search，则 subject 不变，如果定义了可选参数 count，那么该函数只替换 subject 中 count 个 search。下面创建一个示例，具体讲解如何使用这两个函数，如代码 5.24 所示。

代码 5.24　使用 str_replace()函数和 str_ireplace()函数

```
<?php
$str1 = str_replace("world", "body", "hello world world");
echo $str1;// 显示 hello body body
$phrase  = "You should eat fruits, vegetables, and fiber every day.";
$healthy = array("fruits", "vegetables", "fiber");
$yummy   = array("pizza", "beer", "ice cream");
$newphrase = str replace($healthy, $yummy, $phrase, $count);
echo $newphrase;// 显示 You should eat pizza, beer, and ice cream every day.
echo $count; // 显示 3
$bodytag = str_ireplace("%body%", "red", "<body text=%BODY%>");
echo $bodytag;
echo "hello";
?>
```

执行该段代码，结果如下所示：

```
hello body body
You should eat pizza, beer, and ice cream every day.
3

hello
```

9. strstr()函数

strstr()函数通过比较返回一个字符串的部分，语法格式如下所示：

```
string strstr ( string $haystack, string $needle )
```

由该函数的语法格式知，haystack 表示处理的字符串，needle 表示与 haystack 字符串相匹配的项，返回结果为从匹配位置开始到最后的一个字符串，如果没有匹配就返回 false。下面创建一个示例，具体讲解如何使用该函数，如代码 5.25 所示。

代码 5.25　使用 strstr()函数

```
<?php
$name = '唐晓阳';
$domain = strstr($name, '晓');
echo $domain;
?>
```

执行该段代码，结果如下所示：

```
晓阳
```

10. substr()函数

substr()函数功能是获取部分字符串，也可以说是对字符串进行截取，该函数的语法格式

如下所示：

```
string substr ( string $string, int $start [, int $length] )
```

该函数可以有 3 个参数，其中两个参数必须有，string 表示要操作的对象；start 是一个 int 类型的参数，如果是正数表示截取字符串的起始位置，该参数如果为空，默认为 0，如果为负数表示从后往前计数；length 是一个可选的参数，表示截取字符串的长度，如果不使用该参数，截取的字符串一直到最后，如果是负数，表示从后往前计数字符串长度。下面创建一个示例，具体讲解如何使用该函数，如代码 5.26 所示。

代码 5.26　使用 substr()函数

```
<? php
$str ="One World One Dream";
echo substr($str,10);
echo "<br/>";
echo substr($str,10,3);
echo "<br/>";
echo substr($str,10,-5);
echo "<br/>";
echo substr($str,-10);
echo "<br/>";
echo substr($str,-10,4);
echo "<br/>";
echo substr($str,-10,-5);
?>
```

执行该段代码，结果如下所示：

```
One Dream
One
One
One Dream
One
One
```

11．substr_count()函数

substr_count()函数的作用是计算字符串中某字符出现的次数，语法格式如下所示：

```
int substr_count ( string $haystack, string $needle [, int $offset [, int
$length]] )
```

substr_count()函数的第一个参数 haystack 表示被查找的字符串，参数 needle 表示要查找匹配的字符串，参数 offset 表示字符串中开始比较的位置，如果为空或是省略，默认为从头开始。下面使用该函数创建一个示例，如代码 5.27 所示。

代码 5.27　使用 substr_count()函数

```
<?php
$text = 'One World One Dream';
```

```
echo substr_count($text, 'One'); echo "<br/>";
echo substr_count($text, 'One', 3);echo "<br/>";
echo substr_count($text, 'One', 3, 3); echo "<br/>";
?>
```

执行该段代码，结果如下所示：

```
2
1
0
```

12. substr_replace()函数

substr_replace()函数能够替换字符串中部分内容，语法格式如下所示：

```
mixed substr_replace ( mixed $string, string $replacement, int $start [, int
$length] )
```

由该函数的语法格式知，string 参数表示在该字符串中查找并替换，replacement 参数表示要替换的内容，start 参数表示需要处理字符的起始位置，如果该参数为负数，表示替换内容时从后往前计数，length 为可选参数，该参数如果为正数表示替换的长度，负数表示从后往前替换到的位置。下面使用该函数创建一个示例，如代码 5.28 所示。

代码 5.28　使用 substr_replace()函数

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo substr_replace($var, 'Yound', 0); echo "<br/>";
echo substr_replace($var, 'Yound', 0, 0); echo "<br/>";
echo substr_replace($var, 'Yound', 10, -1); echo "<br/>";
echo substr_replace($var, 'Yound', -7, -1); echo "<br/>";
echo substr_replace($var, '', 10, -1);
?>
```

执行该段代码，结果如下所示：

```
Yound
YoundABCDEFGH:/MNRPQR/
ABCDEFGH:/Yound/
ABCDEFGH:/Yound/
ABCDEFGH://
```

5.3　PHP 身份认证

身份认证是 Web 应用程序最常见的做法。有两种身份验证方式比较常用：基本的 HTTP 身份验证和 PHP 身份认证，本节将详细介绍这两种身份验证。

5.3.1　基本的 HTTP 身份验证

HTTP 协议提供了一种非常简单、有效的用户认证方式，就是由服务器来咨询资源请求，客户端用于提供对认证过程至关重要的信息。该认证方式的具体过程如下所示。

（1）客户端请求一个受限资源。

（2）服务器用一个 401（未授权访问）响应消息对这个请求做出响应。

（3）客户端识别 401 响应，弹出一个如图 5-11 所示的认证对话框。如今，许多浏览器都提供了对 HTTP 认证的支持，包括 IE、Netscape Navigator、Opera 和 Mozilla 等。



图 5-11　认证对话框

（4）如果用户提供了相应的用户名和密码，就发送给服务器进行验证，如果通过验证，用户将得到请求访问的资源，否则将拒绝用户访问。

（5）如果用户通过了验证，浏览器将在其认证缓存中存储认证信息。此缓存信息将一直保存在浏览器，直到缓存被清空，或者浏览器收到另一个 401 服务响应。

HTTP 认证可以有效地控制对受限资源的访问，但它不能保证认证信息传输通道的安全。也就是说，对于位置合适的攻击者而言，偷窃或监视服务器与客户端之间发送的数据是很容易实现的。因为用户提供的用户名和密码都包含在这些传输数据中，并且没有经过任何加密。为了解决这种方法可能带来的问题，通常情况下采用安全套接字（SSL）来实现一个安全的通信通道。目前，SSL 得到了所有主流 Web 服务器的支持，包括本书介绍过程中用到的 Apache 和微软的 Internet 信息服务器（IIS）。

5.3.2　PHP 身份认证

PHP 身份认证方式将用户认证直接集成到 Web 应用程序逻辑中。本节将主要介绍 PHP 的内置认证功能，并讲解一些可以立即集成到应用程序中的认证方法。

1．PHP 身份认证基础知识

PHP 使用两个预定义的变量来认证用户：$_SERVER['PHP_AUTH_USER'] 和 $_SERVER['PHP_AUTH_PW']，这两个变量保存了认证需要的两个部分，分别是用户名和密码，但是在使用这两个变量时应注意以下方面。

- 两个变量都必须在受限页面的开始处验证。所以用户可以将认证代码放在单独的文件

中，然后在使用时通过 require()函数将该文件包含进来即可。
- 这两个变量在 CGI 版本的 PHP 中不能正常工作，在 Microsoft IIS 上也不起作用，具体说明读者可查阅相关资源。

另外，PHP 在处理认证时还经常用到两个标准函数：header()和 isset()。下面详细介绍这两个函数。

- **header()函数**

该函数用于向浏览器发送原始的 HTTP 首部，语法格式如下所示：

```
int header ( string string [, bool replace [, int http_response_code]])
```

该函数中的 string 参数表示发送给浏览器的首部信息；可选参数 replace 确定此信息是替换之前发送的首部信息，还是和以前的首部信息一起发送；可选参数 http_response_code 用于定义将随同首部信息一起发送的特定响应码，可以在字符串中包含此相应码。应用于用户认证时，此函数对于向浏览器发送 WWW 认证首部很有用，将显示弹出的认证提示窗口。如果提交了不正确的认证凭证，该函数还能为用户发送 401 首部信息.

- **isset()函数**

该函数用于确定是否已为一个变量赋值，其使用格式如下所示：

```
bool isset ( mixed var [, mixed var [, ...]])
```

由该函数语法格式知，如果变量包含值则返回 true，否则返回 false。应用于用户认证时，该函数用来确定是否正确地设置了$_SERVER['PHP_AUTH_USER']和$_SERVER['PHP_AUTH_PW']变量。

2. 基于认证变量的身份认证

这种方法通过 header()函数发送 HTTP 首部强制进行验证。客户端浏览器则弹出要输入用户名和密码的对话框，用户输入相应的信息并提交，这时用户输入的信息将在被传送到服务端之后保存到$PHP_AUTH_USER，$PHP_AUTH_PW 这两个全局变量中。然后利用这两个变量就可以进行对用户名和密码的验证了。下面创建一个示例，详细讲解如何使用变量的身份认证，如代码 5.29 所示。

代码 5.29 使用变量的身份认证

```
<?php
if(!isset($_SERVER['PHP_AUTH_USER']))
{
    Header('WWW-Authenticate: Basic realm="请输入用户和密码"');
    Header("HTTP/1.0 401 Unauthorized");
    echo "本站提示：你已经取消了登录，本站许多功能你将不能使用";
    exit;
}
else
{
    if ( !($_SERVER['PHP_AUTH_USER']=="admin" && $_SERVER['PHP_AUTH_PW']==
```

```
        "admin") )
        {
        // 如果是错误的用户名称/密码对，强制再验证
        Header(' WWW-Authenticate: Basic realm="请重新输入用户和密码"');
        Header("HTTP/1.0 401 Unauthorized");
        echo "本站提示 : 用户名或密码错误! ";
        exit;
        }
        else
        {
            echo "认证成功，欢迎你登录本站! ";
        }
    }
    ?>
```

在代码 5.29 中，只有$_SERVER['PHP_AUTH_USER']和$_SERVER['PHP_AUTH_PW']分别被设置为 admin 和 admin 后才能取得所请求的资源，否则将强制用户重新输入用户名和密码，或在尝试多次后输出事先设定好的错误消息。保存代码，打开地址栏，在地址栏中输入 http://localhost/001.php，页面效果如图 5-12 所示。

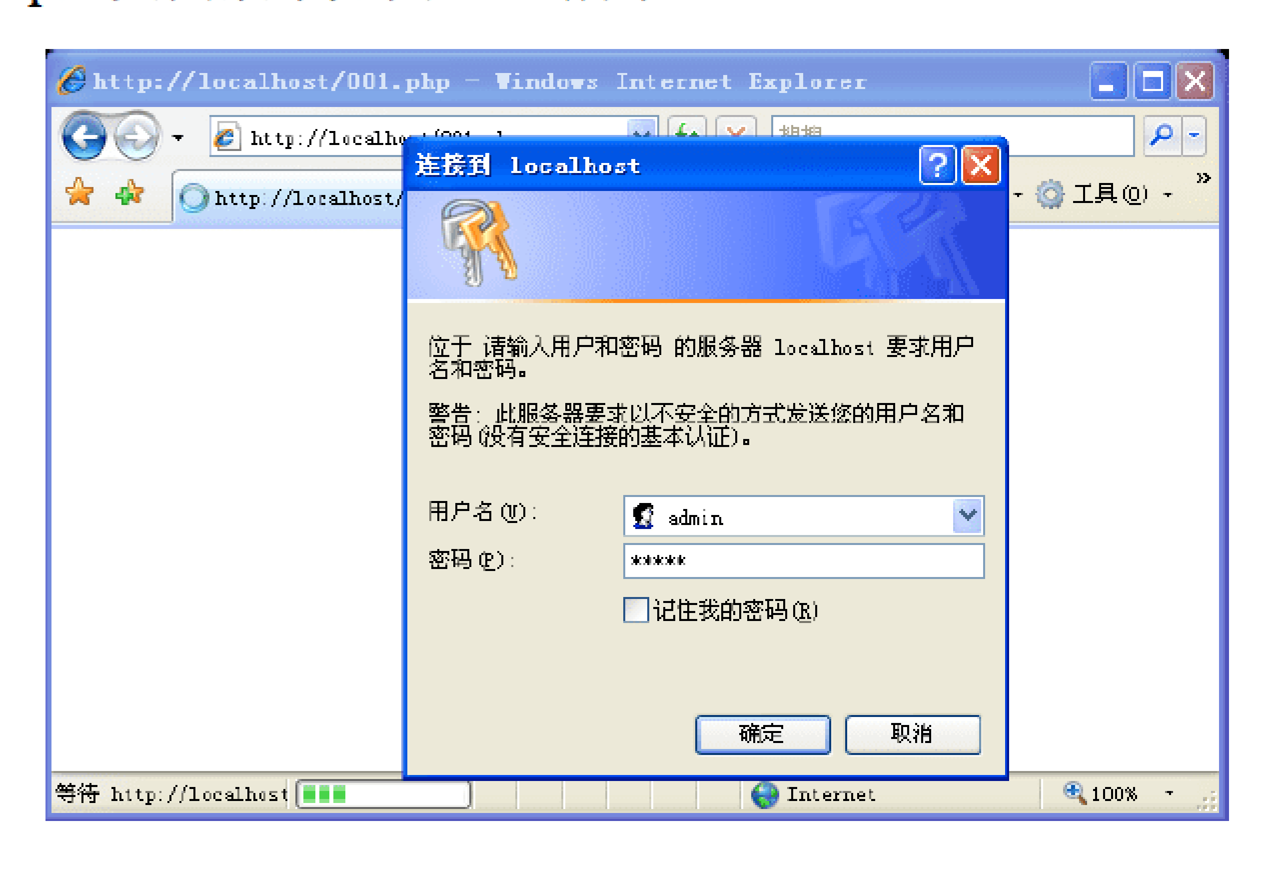


图 5-12　页面效果图

3．基于文件的身份认证

基于文件的身份认证就是利用文本文件为每一位用户提供唯一的登录对，这样就可以记录用户特定的登录时间、活动和动作。其中，该文件的每行包含一个用户名和加密密码对，它们之间用冒号（：）分隔。

这里需要注意的是，该文件应当存储在服务器文档根目录之外。否则，攻击者有可能通过一些不正常手段来获取该文件，从而造成信息的泄露。当然用户可以使用不对称密码，以明文形式存储。但这种方式极不安全，因为如果文件权限配置不当，那么能够访问服务器的用户就可以查看这些登录信息。

基于文件的身份认证通过将文本读取到数组中，然后循环处理数组，搜索匹配的数据来

进行身份认证。在这个过程中需要使用以下函数。

- **file(string filename)** 该函数将文件读取到数组中，数组的每个元素分别包括文件中的一行。
- **explode(string separator,string string [, int limit])** 该函数将字符串分解为一系列字符串，每个字符串的边界由指定的分隔符决定。
- **md5(string str)** 该函数使用 RSA 数据安全公司的 MD5 消息摘要算法（http://www.rsa.com）来计算字符串的 MD5 散列值。

下面创建一个示例，具体讲解基于文件的身份验证。首先新建一个文本文件，在该文件中输入以下内容并保存，将文件重命名为 login.txt。

```
admin:admin
yound:tangyound
dongjielanyu:dongjie
```

新建一个 PHP 文件，该文件用于将用户输入的信息与 login.txt 文件中的登录对进行匹配，如代码 5.30 所示。

代码 5.30　处理用户输入的用户名和密码

```
<?php
$userfile = file("login.txt");
if (!(isset($_SERVER['PHP_AUTH_USER'])) || !(isset($_SERVER['PHP_AUTH_PW']-
)))
    {
      Header('WWW-Authenticate: Basic realm="请输入用户名和密码"');
      Header("HTTP/1.0 401 Unauthorized");
      echo "本站提示: 用户取消了登录! ";
      exit;
    }
    else
    {
      foreach ($userfile as $login)
      {
        list($username, $password) = explode(":", $login);
        $password = trim($password);
        if (($username == $_SERVER['PHP_AUTH_USER']) && ($password == $_SER-
        VER['PHP_AUTH_PW']))
         {
           $authorized = TRUE;
             echo $_SERVER['PHP_AUTH_USER']."身份认证成功，欢迎你".$username;
           break;
        }
         else
         {
           // 如果是错误的用户名称/密码对，强制再验证
```

```
                Header(' WWW-Authenticate: Basic realm="请重新输入用户名和密码"');
                Header("HTTP/1.0 401 Unauthorized");
                echo "本站提示 : 用户名或密码错误! ";
                exit;
            }
        }
    }
?>
```

保存文件，然后打开 IE 浏览器，在地址栏中输入 http://localhost/LoginUser.php，然后单击【转到】按钮，在页面上会显示登录窗口，在登录窗口中输入 login.txt 文件中的任一登录对，这里使用第一行登录对。单击【确定】按钮，成功通过认证后将显示如下所示的信息：

```
admin 身份认证成功，欢迎你 admin
```

如果此时单击【取消】按钮将显示如下所示的信息：

```
本站提示: 用户取消了登录!
```

如果重复输入几次错误的用户名或密码，将显示如下所示的信息：

```
本站提示 : 用户名或密码错误!
```

4．基于数据库的身份认证

当要处理多用户身份认证时，以上两种身份认证方法就不适合了，这时需要实现基于数据库的身份认证。基于数据库的解决方案是功能最强大的一种方法，因为它不仅管理方便，提高了扩展性，还可以集成到更大的数据库基础设施中。基于数据库的身份认证使用所输入的用户名和密码作为查询条件对用户表执行一个查询，如果从数据库中查找到相关的记录，那么该用户通过身份认证，否则给出相关的错误信息。关于该方法的实现读者可以参考后面章节的内容。

5．基于 IP 的身份认证

在基于数据库的身份认证中，由于用户名和密码有可能会被黑客或一些别有用心的人获取。所以为了解决这个问题，进一步保证用户及其信息的安全，一种有效的方法是当用户进行身份认证时，不仅需要合法的用户名和密码登录对，还需要一个与此相关联的 IP 地址。这样即使窃取了用户名和密码，也无法通过身份认证。

基于 IP 的身份认证的实现过程与基于数据库的身份认证的实现过程基本相同，除了使用所输入的用户名和密码作为查询条件外，还需要匹配事先设定的与之相关联的 IP 地址。读者可以对基于数据库的身份认证过程加以改进来实现该方法。

第6章 cookie和会话



在很多网站中，往往需要记录用户的相关信息，以及用户的当前状态。在 PHP 中有一个可行的解决办法，就是在客户端使用 cookie 或 session 保存用户相关信息。当用户再次访问该网站时，可以从用户本地文件中读取相关信息。本章主要讲解 cookie 和会话的相关知识，例如如何定义 cookie、cookie 基本操作和会话等。

学习目标 Objective

- 掌握 cookie 的基本操作
- 熟练控制 cookie 的有效性
- 掌握会话的基本用法
- 理解如何配置 PHP 的会话
- 灵活掌握传输会话 ID 及获得会话 ID
- 掌握如何使用会话存储数据
- 理解会话存储工作原理
- 熟练使用 Session

6.1 cookie

当用户访问网站时，服务器将用户相关信息存储在 cookie 中，并发送到客户端，客户端保存这个 cookie。当用户访问其他页面时，服务器可以从 cookie 中获取保存的用户相关信息。通过使用 cookie 可以克服 HTTP 无状态记录的缺陷，并且可以存储用户信息。在 PHP 中可以通过函数方便地使用 cookie。本节主要介绍如何操作 cookie，以及获取 cookie 中的信息。

6.1.1 cookie 介绍

cookie 是存放在客户端的一组数据，由服务器端的脚本程序生成。PHP 脚本程序可以获取 cookie 中保存的信息，并且 cookie 以文件的形式存放在远程客户端，其作用主要是用来识别用户身份。cookie 在 Web 开发中向来扮演着比较重要的角色，是 Web 开发中历史悠久并且经常使用的技术之一，在 PHP 中体现更为明显，在设计用户身份验证的系统中，往往都会使

用 cookie。

客户端每一项数据都是相互独立的，在需要读取时再分别取出。cookie 就是将数据存放在客户端的技术之一，使用过程如图 6-1 所示。

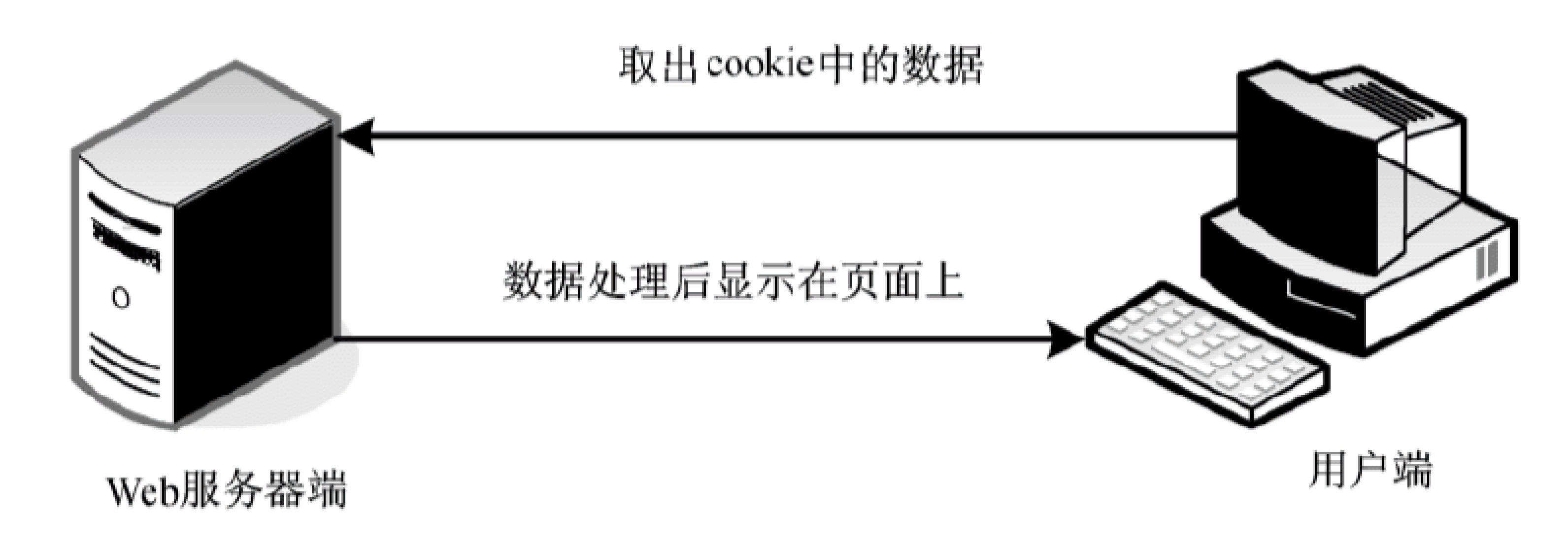


图 6-1　cookie 的使用过程

那么，哪些数据应该存放到客户端的 cookie 中？这样做有什么好处？为何不将这些数据存放在服务器上呢？答案很明显，如果将数据存放在服务器上，那么管理起来就比较麻烦。有些数据就一定要存放在服务器中，例如用户的账号名称、密码等。不过有些数据并不是那么重要，例如用户昵称和用户访问本站的次数等，如果将这些信息也都存放在服务器中，撇开数据储存的容量不说，光是搜索庞大的数据，就要花费不少时间，这将对服务器的性能产生影响。如果能将这些个人数据存放在用户计算机中，将可大大减轻服务器的负担。

大部分的 Web 程序开发者都会利用 cookie 来储存不太重要的用户个人数据，而将重要的认证数据，例如交易数量、客户 ID 和密码等储存在服务器中。

6.1.2　基本操作

即使会话结束 cookie 也能在客户端保存用户的相关信息，因此可以在后续会话中读取。从这方面来看，cookie 在会话中起到了很重要的作用，本节将介绍如何设置创建 cookie，以及一些对 cookie 的基本操作。

1．设置 cookie

cookie 是 HTTP 协议头的一部分，用于浏览器和服务器之间传递信息，所以必须在任何属于 HTML 文件本身的内容输出之前调用 cookie 函数。在 PHP 中，用 setcookie()函数来设置 cookie，该函数的语法格式如下所示：

```
bool setcookie ( string name [, string value [, int expire [, string path [,
string domain [, int secure]]]]])
```

由函数语法格式知，setcookie()函数的所有参数中除了 name 参数外都是可选项。如果只有一个 name 参数，则表示删除客户端浏览器中名字为 name 的 cookie。也可以通过设置为空字符串来跳过一个参数，但 expire 和 secure 的值为整数，不能用空字符串，只能用 0 来代替。expire 参数是一个由 time()或者 mktime()函数返回的 UNIX 格式的函数。secure 表示 cookie 只在建立 HTTP 连接时才发送。

创建好的 cookie 不会立即生效，而是等到请求下一个页面时才能生效。这是由于设置好的 cookie 由服务器传递给客户端，在请求下一个页面时，客户端才能将 cookie 取出并传回服务器。

下面创建几个使用 cookie 的示例，具体代码如下所示：

```
Setcookie("Mycookie", "Value of Mycookie");//简单地创建 cookie
Setcookie("WithExpire", "Expire in 1 hour", time()+3600);
                                              //创建带有失效时间的 cookie
Setcookie("Fullcookie",  "Full  cookie  value",  time()+3600,  "/forum",
".phpuser.com", 1);                           //创建全面的 cookie
```

Setcookie()函数里表示 value 的部分，在传递时会自动被 encode，也就是说，如果 value 的值是“test value”，在传递时就变成了“test%20value”，跟 URL 的方法一样。当然，对于程序来说这是透明的，因为在 PHP 接收 cookie 的值时会自动将其 decode。如果要设置同名的多个 cookie 可使用数组，下面创建一个同名的多个 cookie 示例，具体代码如下所示：

```
Setcookie("cookieArray[]", "Value 1");
Setcookie("cookieArray[]", "Value 2");
或
Setcookie("cookieArray[0]", "Value 1");
Setcookie("cookieArray[1]", "Value 2");
```

2. 接受和处理 cookie

PHP 对 cookie 的接受和处理非常容易，这是因为 PHP 可以自动将创建好的 cookie 值作为一个 PHP 变量使用，该变量具有与这个 cookie 相同的名称。例如设置一个名为 Mycookie 的 cookie，PHP 会自动从 WEB 服务器接收的 HTTP 头里将其分析出来，并形成一个与普通变量一样的变量，名为$Mycookie，这个变量的值就是 cookie 的值。数组同样适用。另外一个办法是引用 PHP 的全局变量$HTTP_COOKIE_VARS 数组。下面创建一个示例具体说明在 PHP 中如何接受和处理 cookie，如代码 6.1 所示。

代码 6.1　接受和处理 cookie

```
<?php
setcookie("cookieName1","cookievalue1",time()+3600);
setcookie("cookieName2","cookieValue2",time()+3600);
echo $_COOKIE["cookieName2"]."<br/>";
print_r($_COOKIE);
?>
```

执行代码 6.1，结果如下所示：

```
cookieValue2
Array
(
    [cookieName1] => cookievalue1
```

```
    [cookieName2] => cookieValue2
)
```

3．删除 cookie

由于 cookie 有一个过期时间，所以在创建之后的某个时刻，将被自动删除。不过用户也可以立即删除一个 cookie。要删除一个已经存在的 cookie 有两个办法：一是调用只带有 name 参数的 setcookie()，那么名为这个 name 的 cookie 将被从用户计算机上删掉；另一个办法是设置 cookie 的失效时间为 time()或者 time()–1，那么这个 cookie 在这个页面浏览完之后就被删除了，其实是失效了。例如，要删除前面创建的 cookieName1，可以使用如下所示的方法：

```
setcookie("cookieName1 ","",time()-1800);
setcookie("cookieName1");
```

下面创建一个示例，对前面访问 cookie 的示例进行修改，来展示如何删除 cookie，如代码 6.2 所示。

代码 6.2　删除 cookie

```
<pre>
<?php
    setcookie("admin","cookieValue1",time() + 3600);
    setcookie("administrator","cookieValue2",time() + 3600);
    echo $_COOKIE["admin"]."\n";
    echo $_COOKIE["administrator"]."\n";
    setcookie("admin","",time() - 1800);
    setcookie("administrator");
    print_r($_COOKIE);
?>
</pre>
```

成功执行代码 6.2，将输出如下所示的信息：

```
cookieValue1
cookieValue2
Array
(
    [admin] =>
    [administrator] =>
)
```

4．创建多值 cookie

在 PHP 中，一个 Web 站点只能在用户系统上存储 20 个 cookie，如果能够在一个单独的 cookie 中存储多个值将十分有用。要想做到这一点，可以将值放在一个数组中，如果用户愿意还可以使用 serialize()函数将数组的元素序列化到一个字符串中，用户还可以使用

unserialize()函数恢复数组的值。下面创建一个实例演示如何创建和访问包含多个值的 cookie，具体如代码 6.3 所示。

代码 6.3　创建和访问包含多个值的 cookie

```
<?php
    for($i=0;$i<15;$i++)
    {
        $array[$i] = $i;
    }
    echo "<li>显示创建好的数组信息:</li><hr/>";
    echo "<fieldset><legend>数组详细信息</legend>";
    print_r($array) ;
    echo "</fieldset>";
    echo "<br/>";
    //========创建 cookie=============
    $s = serialize($array);
     setcookie("cookies",$s);
    echo "<li>显示 cookie 保存的信息:</li><hr/><fieldset>
          <legend>cookie 值信息</legend>";
    if(isset($_COOKIE["cookies"]))
    {
     $array = unserialize(stripslashes($_COOKIE["cookies"]));
        foreach($array as $i => $cookie)
       {
           echo $i."=>".$cookie."…^…";
       }
    }
    echo "</fieldset>";
?>
```

保存代码 6.3，然后在浏览器中打开该 php 页面，页面效果如图 6-2 所示。

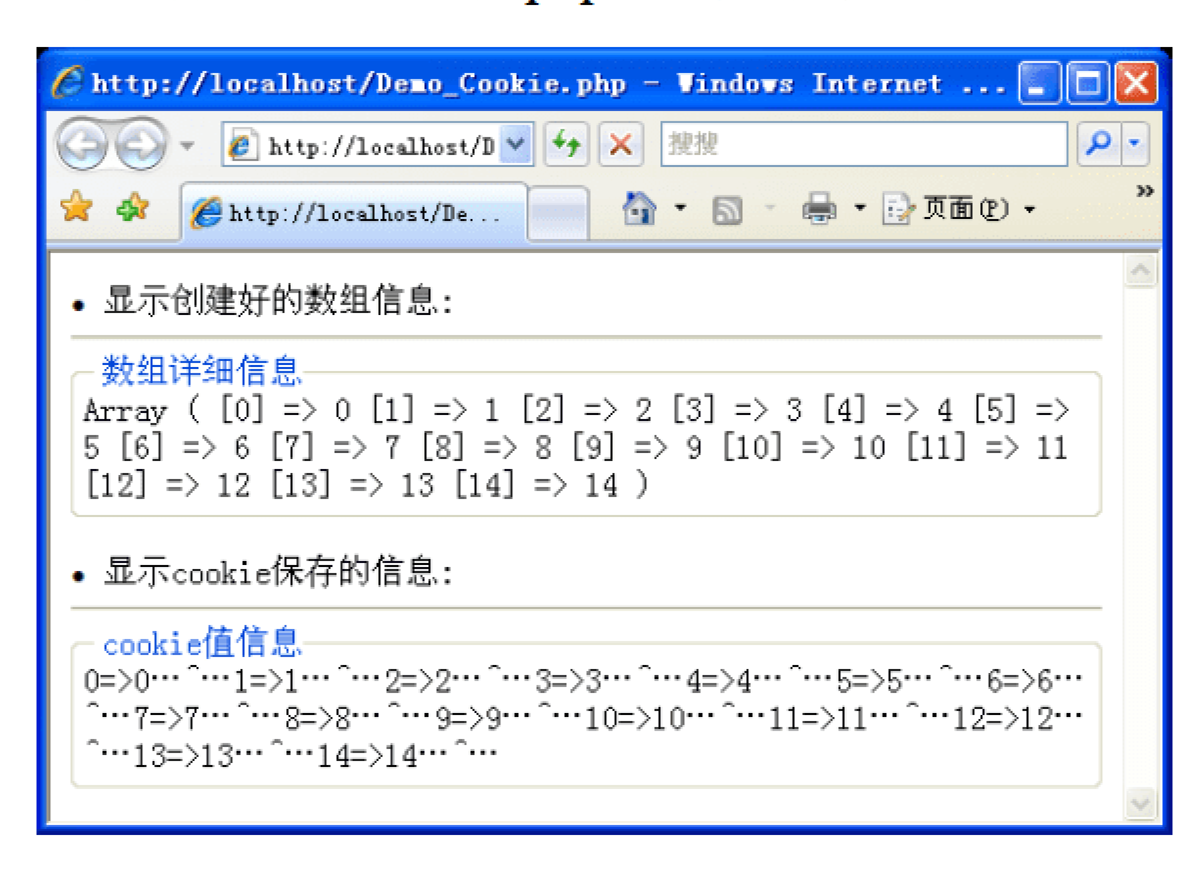


图 6-2　页面效果图

5. 使用 cookie 的限制

在网站中使用 cookie，首先必须在 HTML 文件的内容输出之前设置；其次不同的浏览器对 cookie 的处理也不一致，且有时会出现错误的结果；第三个限制是在客户端，一个浏览器能创建的 cookie 数量最多为 30 个，并且每个不能超过 4KB，每个 Web 站点能设置的 cookie 总数不能超过 20 个。

6.1.3　cookie 有效性控制

通过上面对 cookie 的学习，读者对 cookie 有了一定的了解。在定义 cookie 时，可以定义 cookie 的有效期。由于 cookie 保存在客户端，有效控制 cookie 的有效期，可以增强网络用户的信息安全。本节主要讲解动态控制 cookie 有效期。

一个合理的有效期，可以最大限度地保护用户相关信息。下面创建一个示例，通过一个登录页面具体讲解如何动态设置 cookie 有效期。首先创建一个登录页面，如代码 6.4 所示。

代码 6.4　用户登录页面

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> cookie 有效性控制</title>
</head>
<body>
<center>
<table width="400" border="0">
  <tr>
    <td>
    <fieldset>
      <legend>用户登录页面</legend>
     <form action="savecookie.php" method="post">
      <p>用户名:
        <input type="text" name="username"></p>
      <p>密  码:
        <input type="password" name="pwd"></p>
       <p>cookie 有效期:
        <label>
        <select name="savetime">
        <option value="1800">半个小时</option>
        <option value="3600">一个小时</option>
        <option value="86400">一天</option>
        <option value="604800">一周</option>
        <option value="2592000">一个月</option>
        </select>
```

```
          </label>
         </p>
        <input type="submit" value="提    交" >
      </form>
       </fieldset>
      </td>
    </tr>
  </table>
  </center>
  </body>
  </html>
```

在代码 6.4 中，创建了一个用户登录的页面，用户可以在该页面上输入用户的相关信息，并且可以选择 cookie 保存的时间，效果如图 6-3 所示。



图 6-3　用户登录页面效果图

下面创建保存 cookie 的页面 savecookie.php，首先在该页面上获取 cookie 信息，然后创建 cookie，并且在该页面判断用户是否输入信息，如代码 6.5 所示。

代码 6.5　处理 cookie 代码

```
<?php
  $username = $_POST['username'];
  $pwd = $_POST['pwd'];
  $savetime = $_POST['savetime'];
  setcookie("usernamecookie",$username,time()+$savetime);
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>通过传入的时间设置 cookie 的有效期</title>
</head>
<center>
<?php
  if($username == "" OR $pwd == "")
```

```
    {
     echo "请输入用户名称或密码! ";
    }
     else
     {
            echo "cookie 数据已经顺利存储<hr/><p>";
            echo "目前\$usernamecookie 是: [".$username;
            echo "]有效期是: ".$savetime."秒</p>";
      }
echo "<br/><a href='Demo-login.php'>重新输入</a>";
?>
</center>
<body>
</body>
</html>
```

保存代码 6.5，在浏览器中打开用户登录页面，在该页面上输入相关信息，并单击【提交】按钮转到保存 cookie 页面，页面效果如图 6-4 所示。

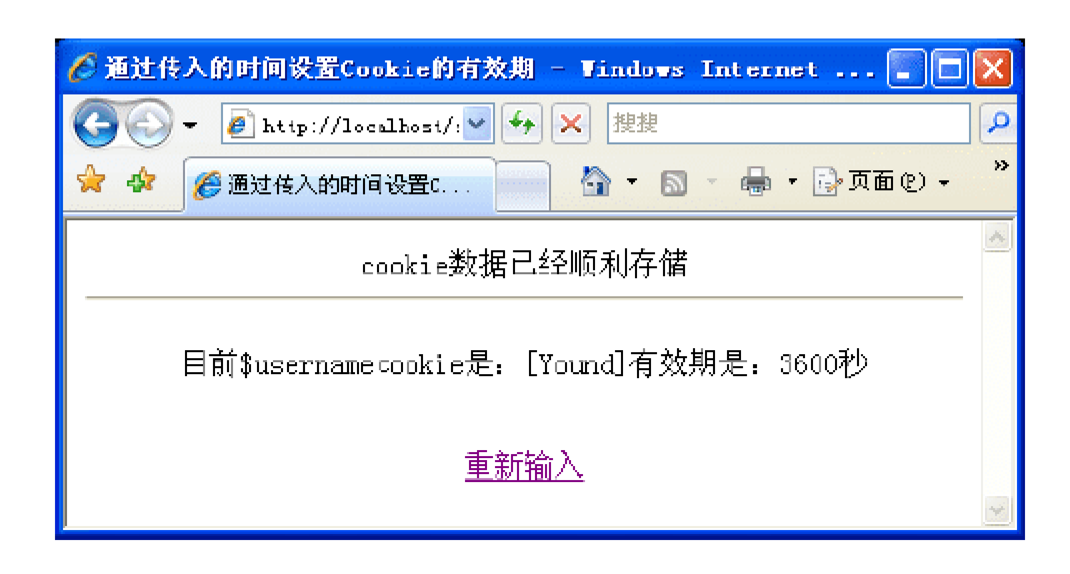


图 6-4　页面效果图

在本示例中，通过手工设置 cookie 的有效期来控制 cookie 的有效性。这样 cookie 就会按照设置的时间失效，从而有效地保护用户信息的安全。

6.2　会话

会话（session）是为特定用户识别和管理状态的一种方式。当发生会话时，每位网站访问者都会得到一个会话 ID（SID），然后将此会话 ID 与任意数量的数据关联。服务器使用会话 ID 在处理请求之前定位相应的会话。本节主要介绍会话基本用法，会话所需要的配置和会话工作原理等。

6.2.1　PHP 会话配置

在 PHP 中有许多配置指令负责确定 PHP 会话处理功能的行为，这些配置指令在确定这种行为时会起到很重要的作用。本节将介绍比较重要的一些配置指令。

1. session.save_handler(files ,mm,sqlite,user)

该指令用于确定如何存储会话信息，可以通过以下 4 种方法之一进行存储：文件（files）、共享内存（mm）、SQLite 数据库（sqlite）和用户自定义函数（user）。其中，files 选项会话存储文件的数量可能会达到几千个，甚至在一段时间后会达到几十万；共享内存选项速度最快，但却最不可靠，因为数据存储在 RAM 中；sqlite 选项利用了新的 SQLite 扩展，通过这个轻量级数据库透明地管理会话信息；用户自定义选项虽然配置复杂，但也是最灵活、功能最强大的一个选项，因为可以创建定制处理函数，在开发人员所需的任何媒体上存储信息。

2. session.save_path (string)

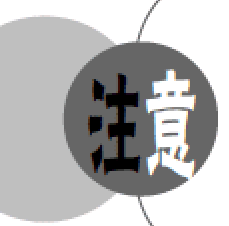
该指令与 session.save_handler 指令结合使用，如果 session.save_handler 设置为 files 存储选项，则该指令必须指向存储目录。这里需要注意的是，该指令不能设置为位于服务器文档根目录下的某个目录，否则入侵者可以轻松地通过浏览器危害这些信息的安全。另外，所指向的存储目录必须是服务器守护进程可写的。通常情况下，出于效率方面的原因，可以使用 N;/path 来定义 session.save_path，其中 N 是一个整数，表示可以存储会话数据的 N 层深度子目录的数量。该指令的作用域：PHP_INI_ALL；默认值：/tmp。

3. session.name (string)

该指令用于确定 cookie 名，其作用域：PHP_INI_ALL；默认值：PHPSESSID。这里可以将默认值改为更适合于应用程序的名字，或者可以通过 session_name()函数在需要时进行修改。

4. session.auto_start (boolean)

该指令用于确定是否自动启动会话，其作用域：PHP_INI_ALL；默认值：0。如果将该指令设置为 1，则自动启动会话；而通过函数 session_start()可以显式启动会话。如果用户要在网站中一直使用会话，可以考虑设此指令为 1，否则可以设置为 0，在必要时调用 session_start()函数启动会话。



启动该指令将无法在会话中存储对象，因为类定义要在会话开始之前加载才能重新创建对象。但 session.auto_start 会禁止这一点，所以如果要在会话中管理对象，就要禁用该指令。

5. session.serialize_handler (string)

该指令定义了串行化和逆串行化数据所用的回调处理器，其作用域：PHP_INI_ALL；默认值：php。在默认情况下，由 PHP 的内部处理器来处理。PHP 还支持另一个串行化处理器，即 Web 开发数据交换（WDDX），在提供 WDDX 支持的条件下编译 PHP 就可以使用这个串行化处理器。

6. session.gc_probability (integer)

该指令用于定义所有概率的分子部分，而该分子部分用于计算调用垃圾回收过程的频率。

7．session.gc_maxlifetime (integer)

该指令用于确定有效会话的持续时间，以秒为单位，其作用域：PHP_INI_ALL；默认值：1440（默认 24 分钟，即 1440 秒）。当到达此限制时，会话信息将被销毁，系统资源得以重新分配。

8．session.referer_check (string)

该指令用于指定一个用于验证每位用户的子串，如果没有包含此子串，会话 ID 将失效。如果使用 URL 重写来传播会话 ID，很多人都能通过复制和散布 URL 来查看某个会话状态，使用该指令可以减少这种可能性。

9．session.entropy_file (string)

该指令用来指向一个额外的信息源（Entropy Source），并将这个信息源集成到会话 ID 生成过程中。在 Windows 系统下，安装 Cygwin（http://www.cygwin.com/）会提供类似于 random 或 urandom 的功能。而在 UNIX 系统下，这个来源通常是/dev/random 或/dev/urandom。

10．session.entropy_length (integer)

该指令确定从 session.entropy_file 所指定文件中读取字节数。如果 session.entropy_file 为空，则此指令被忽略，并使用标准的会话 ID 生成模式。

11．session.use_cookies (boolean)

该指令设置为 1，表示使用 cookie 进行会话 ID 传播；设置为 0 将使用 URL 重写。其作用域：PHP_INI_ALL；默认值：1。如果要在多次访问网站之间维护用户会话，就应当使用 cookie，这样处理器可以重新得到会话 ID，继续原来所保存的会话。如果用户数据只用于一次访问，就可以使用 URL 重写。

12．session.use_only_cookies (boolean)

该指令用来确定只使用 cookie 来维护会话 ID，而忽略通过 URL 传递会话 ID 的尝试。其作用域：PHP_INI_ALL；默认值：0。设置此指令为 1 时，PHP 只使用 cookie，设置为 0 则同时使用 cookie 和 URL 重写。

13．session.cookie_lifetime (integer)

该指令用于确定会话 cookie 的有效期，以秒为单位。其作用域：PHP_INI_ALL；默认值：0。如果设置该指令为 0，则 cookie 将一直存活到浏览器重新启动。例如，要使 cookie 存活 1 小时，就将该指令设置为 3600。

14．session.cookie_path(string)

该指令用来确定 cookie 在哪个路径中有效。对于这个有效路径下的所有子目录，此 cookie

同样有效。例如，如果该指令设置为/，则 cookie 对整个网站都有效。如果设置为/bbs，则只有在 http://www.itzcn.com/bbs/路径中调用 cookie 才有效。

15．session.cookie_domain (string)

该指令确定 cookie 在哪个域中有效，其作用域：PHP_INI_ALL；默认值：空。该指令是必须的，因为它能防止其他域读取用户的 cookie。该指令的设置如下所示：

```
session.cookie_domain = www.itzcn.com;
```

如果要在网站子域中使用会话，假如现在有 bbs.itzcn.com、books.itzcn.com 和 download.itzcn.com 子域，该指令可以如下所示设置：

```
session.cookie_domain = .itzcn.com;
```

16．session.cookie_secure (boolean)

该参数设置 cookie 的安全标记，这样可以防止浏览器通过不加密的连接发送会话 cookie。当该参数设置为 on 时，浏览器通过一个使用 SSL（安全套接字层）保护的网络连接发送会话 cookie。其作用域：PHP_INI_ALL；默认值：off。默认值 off 允许浏览器通过加密和不加密的服务发送会话 cookie。

17．session.cache_limiter (string)

该指令用来确定会话页面是否被缓存及如何缓存。它可以被设置为以下 5 个值之一。

- **nono** 该设置禁止随启用会话的页面传输任何缓存控制首部。
- **nocache** 该设置确保对于每个请求，在可能提供缓存的版本之前，先将请求发送到最初的服务器。
- **private** 该设置指定缓存的文档是私有的，该文档只用于最初的用户，不能与其他用户共享。
- **private_no_expire** 该设置是 private 的变体，它保证不会向浏览器发送任何文档过期日期。
- **public** 该设置认为所有文档都是可缓存的，最初的文档请求可能需要认证。

18．session.cache_expire (integer)

该指令确定在创建新页面之前缓存的会话页面可用的时间（以秒为单位）。其作用域：PHP_INI_ALL；默认值：180。如果 session.cache_limiter 设置为 nocache，则该指令将被忽略。

19．session.use_trans_sid (boolean)

该指令被启用后，可以消除使用 URL 重写过程中发生错误的可能性。其作用域：PHP_INI_SYSTEM | PHP_INI_PERDIR；默认值：0。如果禁用 session.use_cookies，为了确保传播 ID，用户的会话 ID 必须增加到 URL。用户可以手工将变量$SID 追加到每个 URL 的后面，也可以通过启用指令自动处理。

20．url_rewriter.tags (string)

该指令的作用域：PHP_INI_ALL；默认值：a=href,area=href,frame=src,input=src,form=fakeentry。启用 session.use_trans_sid 时，在将文档发送给客户端之前，会话 ID 会自动追加到所请求文档中的 HTML 标记后面。但是，与超链接或表单标记不同，许多标记在启动服务器请求时不起作用，所以可以使用 url_rewriter.tags 指令告诉服务器会话 ID 应当追加到哪些标记后面。指令的使用类似如下所示：

```
url_rewriter.tags a=href,frame=src,form=,fieldset=
```

6.2.2　会话工作原理

session 是一种保存上下文信息的机制。针对每一个用户，变量的值保存在服务器端，通过 SessionID 来区分不同的用户。session 以 cookie 或 URL 重写为基础，默认使用 cookie 来实现。系统会创建一个名为 JSESSIONID 的输出 Cookie，或称为 Session Cookie，Cookie 存储在浏览器中，并不写在硬盘上。但是将浏览器的 cookie 禁止后，使用 response 对象的 encodeURL 或 encodeRedirectURL 方法编码 URL，Web 服务器会采用 URL 重写的方式传递 SessionID，用户就可以在地址栏上看到 jsessionid=A09JHGHKHU68624309UTY84932 之类的字符串。通常 Session Cookie 不能跨窗口使用。当用户新开了一个浏览器进入相同的页面时，系统会赋予用户一个新的 SessionID，这样就达不到信息共享的目的。此时可以将 SessionID 保存在 Persistent Cookie 中，然后再从新的窗口中读出来，就可以得到上一个窗口的 SessionID 了。这样通过 Session Cookie 和 Persistent Cookie 的结合，就实现了跨窗口的会话跟踪。

在 PHP 中每一个 Session 都通过调用 session_start()函数开始这个函数检查一个 Session 是否存在，如果不存在则创建一个新 Session。用于演示 Session 工作原理的标准例子之一就是页面计数器。这是一个简单的基于 Session 的计数器，在用户第一次访问一个 Web 页面时初始化一个变量，当每一次用户重新装入这个页面时，该变量的值会自增 1。具体如代码 6.6 所示。

代码 6.6　页面计数器

```
<?php
//==创建 session 变量开始一个会话===
session start();
//$_SESSION['counter'];
session_register('counter');
//增加计数器
$counter=$_SESSION['counter']+1;
$_SESSION['counter']=$counter;
echo  "第".$_SESSION['counter']."次访问本页面！ ";
?>
```

执行该段代码，首次访问该页面执行结果如下所示，如果再次访问该页面，页面计数器

会自动增加访问次数。

```
第 1 次访问本页面!
```

6.2.3 基本用法

PHP 会话通过文件保存，会话文件一般都很小，但是文件数目却很多。本节将介绍一些关键的会话处理任务，并给出相关的会话函数。这些任务包括创建和删除会话，指派和获取会话 ID。

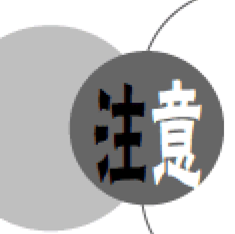
1. 开始会话

超文本传输协议定义了通过万维网传输文本、图形、视频和所有其他数据所用的规则，并且该协议是一种无状态的协议。每次请求的处理都与之前或之后的请求无关，这就导致 HTTP 不能记忆用户过去和将来的环境。因此，需要对每次请求显示地启动和恢复会话。在 PHP 中，可以使用 session_start()函数实现记忆和启动恢复会话功能，该函数的语法格式如下所示：

```
bool session_start ( void )
```

session_start()函数创建一个新会话或者继续当前会话，这取决于是否拥有会话 ID。开始会话时，只需如下调用 session_start()函数：

```
session_start();
```



无论结果如何，session_start()函数都会报告成功的结果。因此，使用任何异常处理都不起作用。如果启用了配置指令 session_auto_start，不必执行该函数，就可以开始会话，但是这样一来每次启用 PHP 页面都会开始或继续一个会话。

2. 创建会话变量

以往程序员经常会使用 session_rigister()函数创建会话变量。在一个会话初始化之前，调用 session_start()函数打开会话文件，并且通过 session_rigister()函数创建会话变量，具体示例如下所示：

```
session_register("add");
$add = "http://www.itzcn.cn";
```

上述语句中变量$add 与其值被自动保存在会话存储器中。这里应该注意，将变量的名称而不是变量本身传递给 session_register()函数。一旦创建，会话变量就变得持久并对初始化会话的脚本可用。PHP 追踪会话变量的值并将值保存到会话文件，所以不要在脚本结束之前显式地保存会话变量。

首先创建会话变量的方法是像其他任何变量一样设置，不过需要使用超级全局变量 $_SESSION 在上下文中引用这些变量。假设希望创建名为 admin 的会话变量，具体语句如下

所示：

```
<?php
    session_start();
    $_SESSION['admin'] = "Yound Tang";
    echo "The admin is:".$_SESSION['admin'];
?>
```

成功执行上述语句，将输出如下所示的信息：

```
The admin is:Yound Tang
```

3. 删除会话变量

用户可以使用 unset()删除会话变量。下面修改创建会话变量的示例，具体如下所示：

```
<?php
    session_start();
    $_SESSION['admin'] = "Yound Tang";
    echo "The admin is:".$_SESSION['admin']."<br/>";
    unset($_SESSION['admin']);
    echo "Now The admin is:".$_SESSION['admin']."<br/>";
?>
```

成功执行上述语句，将输出如下所示的信息：

```
The admin is:Yound Tang
Now The admin is:
```

4. 撤销会话

有时可能让用户在离开网站时注销登录，或者当用户点击适当的链接时，可以从内存中消除当前会话中的所有变量，甚至从存储机制中完全消除整个会话，从而实现手工破坏会话。这些功能可以通过使用 session_unset()函数和 session_destroy()函数完成。

- **session_unset()函数**

该函数的具体使用格式如下所示：

```
void session_unset ()
```

该函数用于清除存储在当前会话中的所有变量，能有效地将会话重置为创建时的状态（没有注册任何会话变量的状态）。需要注意的是，该函数并非从存储机制中完全删除会话。如果希望完全删除会话，需要使用函数 seesion_destroy()。

- **session_destroy()函数**

该函数的具体使用格式如下所示：

```
bool session_destroy ()
```

该函数用于从存储机制中完全删除会话，使当前会话失效。但该函数不会销毁用户浏览

器中的任何 cookie。如果用户不想在会话之后使用 cookie，只需要在 php.ini 文件中将 session.cookie_lifetime 设置为 0。关于这个函数的具体使用如代码 6.7 所示。

代码 6.7　完全删除会话

```
<?php
  // 只在请求设置了一个$PHPSESSID 时才试图结束会话。
  if(isset($PHPSESSID)) {
    $message = "<p/>End of session ($PHPSESSID).";
    session_start();
    session_destroy();
  } else {
    $message = "<p/>There was no session to destroy!";
  }
?>
<html>
  <head><title>Sessions</title></head>
  <body>
    <?=$message?>
  </body>
</html>
```

6.2.4　获取会话 ID

在 PHP 脚本第一次调用 session_start()时，将产生一个会话 ID(SID)，该 ID 将所有的会话数据绑定到某个特定用户。虽然 PHP 能够自动创建和传播会话 ID，但并不是所有的时候都希望使用 PHP 自动创建和传播 ID，有时也需要手工获取和设置会话 ID，这就需要用到 session_id()函数，该函数的语法格式如下所示：

```
string session_id ( [string id])
```

在使用该函数时，如果没有参数，则返回当前会话 ID。如果包括可选参数 id，当前会话 ID 将被该值替换。下面创建一个示例，具体讲解如何使用 session_id()函数，代码如下所示：

```
<?php
session_start();
echo "PHP 自动产生的 SID: ".session_id();
echo "<hr/>";
session_id('youndTang');
echo "手工编写的 SID: ".session_id();
?>
```

执行该段代码，结果如下所示：

```
PHP 自动产生的 SID: okn3edkfjjrjvp5c8ileml6q07
----------------------------
```

```
手工编写的 SID: youndTang
```

6.2.5　会话 ID 传输

由前面章节的介绍，读者知道 PHP 脚本第一次调用 session_start()时，将产生一个会话 ID，并且默认情况下会在响应中包含一个 Set-cookie 首部字段。响应以名称 PHPSESSID 和会话 ID 的值在浏览器中设置一个会话 cookie。PHP 会话管理自动地包含该 cookie，而不需要调用 setcookie()或 header()函数。会话 ID 是 32 个十六进制数字的随机字符串，与其他 cookie 一样，会话 ID 的值在$HTTP_COOKIE_VARS 关联数组和$PHPSESSID 变量中对 PHP 脚本可用。

当开始一个新会话时，PHP 创建一个会话文件。在默认配置下，会话文件使用会话 ID 加前缀 sess_作为文件名写到/tmp 目录中。但会话 ID 是如何传输的呢？主要使用以下两种方法。

- **使用 cookie**

用户在访问网站时，会在必要时获取会话 ID，并在页面中使用与会话 ID 相关的各项数据。这样，即使会话结束，cookie 也能在客户端保存，所以可以在后续会话中读取。但这里要注意的是，由于用户可以通过浏览器控制是否接受 cookie，所以在进行程序设计时要考虑到这方面的问题。

- **使用 URL**

用户可以通过在每个请求页面的本地 URL 上增加会话 ID 来实现其传输。无论用户何时单击这些本地链接，都会导致会话 ID 的自动传播，这里所采用的方法被称为 URL 重写。这样即使客户端禁用 cookie，也不会影响网站会话处理功能的正常使用。但是，这种方法也存在一定的问题。首先，URL 重写在会话之间不能保证持久性，因为一旦用户离开网站，向 URL 自动追加会话 ID 的过程就无法再持续。其次，无法阻止用户将 URL 复制到电子邮件并发送给另一位用户，并且只要会话没有超时，会话就会在接收者的计算机上持续。如果两个用户同步使用相同的会话，或者链接的接收者无意中看到了会话的秘密数据，都有可能发生严重灾难。所以出于安全方面的考虑，建议使用基于 cookie 的方法，但具体如何实现读者，可以根据自己的项目进行决定。

如果调用 session_start()，并且请求包含 PHPSESSID cookie，PHP 将试图查找会话文件并初始化关联的会话变量。但是，如果识别的会话文件未能找到，session_start()将创建一个空的会话文件。

不要将$PHPSESSID 的存在用作新会话的标识，或者用作访问会话 ID 的方法。在第一次调用脚本和创建会话时，PHPSESSID cookie 可能还没有设置，只有后续的请求能确保包含 PHPSESSID cookie。PHP 中提供了 session_id()函数，用于返回初始化会话的会话 ID。

6.2.6　会话数据

会话存储数据就是使用创建的会话变量存储数据，所以会话变量的类型决定了会话要存储的数据。会话变量可以是布尔型、整型、双精度型、字符串型、对象型，或者是这些变量类型的数组。使用对象会话变量时必须小心，因为当初始化一个已有会话时，PHP 需要访问

已注册对象的类定义。如果对象被保存为会话变量，用户应该将这些对象的类定义保存在初始化会话的所有脚本中，而不管脚本是否使用类。

PHP 通过将值串行化（serialize），而将会话变量保存在会话文件中。变量的串行化表示将名称、类型和值保存为一个适合于写到文件的字符流，如下所示：

```
count|i:6;start|i:986084651
```

在下面使用会话存储数据的示例中，创建了两个变量：一个是整型的$count，每调用一次该变量就自动增加 1；另一个是$start，当会话第一次初始化时，就设置为从库函数 time()返回的当前时间。示例中通过测试会话变量$count 是否已注册来确定是否创建了一个新的会话。如果变量$count 还没有注册，将自动增加 1。所以，$count 用于显示脚本被调用了多少次，而 time()-$start 用于显示会话持续了多少秒。具体如代码 6.8 所示。

代码 6.8　使用会话存储数据

```
<?php
session_start();
//如果这是一个新的会话，那么将不注册变量$count
if(!session_is_registered("count"))
  {
     session_register("count");
     $_SESSION['start'] = time();
     $count = 0;
  }
 else
 {
     $count++;
 }
   $sessionId = session_id();
  echo  "Session Indentification Number is :".$sessionId;
  echo "<br/>count=".$count;
  echo "<br/>session 变量开始时间=".$_SESSION['start'];
  $dur = time() - $_SESSION['start'];
  echo "<br/>该 session 值持续了".$dur."秒";
?>
```

打开 IE 浏览器，执行该段代码，单击【转到】按钮将会显示如图 6-5 所示的效果。

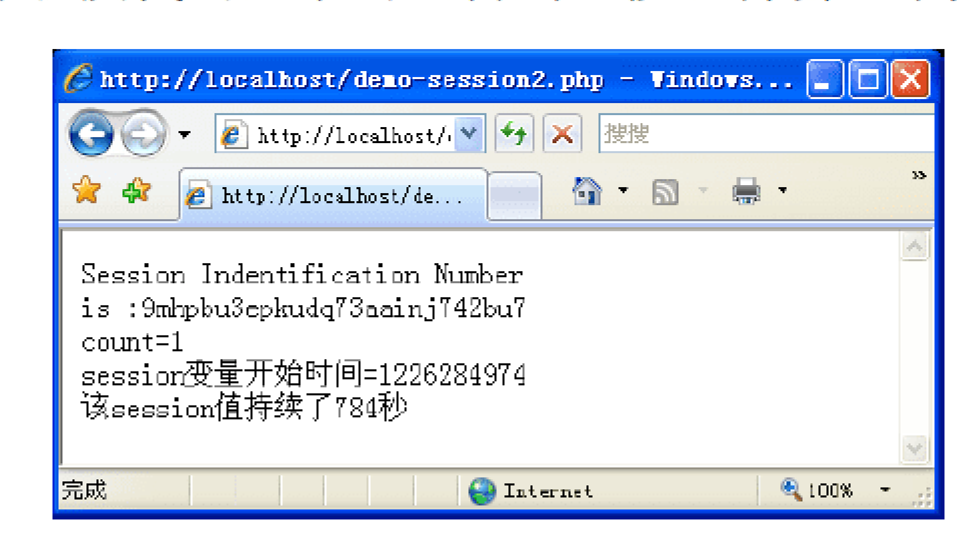


图 6-5　页面最终效果图

6.2.7　编码和解码会话数据

在 PHP 中以标准化格式存储会话数据，并且各个会话变量之间都以分号隔开，每个会话由 3 部分组成：名、长度和值，存储会话数据的语法格式如下所示：

```
"Name|s:lengthvalue:"value"
```

下面有两个变量，并且这两个变量在 PHP 会话中存储，代码如下所示：

```
admin|5:"yound";pwd|6:"youndTang"
```

在 PHP 中，不仅可以标准格式化地存储会话数据，还可以自动处理会话的编码和解码；还可以使用 session_encode()和 session_decode()函数手工完成会话数据的编码和解码。下面分别详细介绍这两个函数。

1．session_encode()编码函数

该函数可以使用手工将所有会话变量编码为一个字符串，然后将此字符串插入到数据库，并在获取后使用解码函数 session_decode()函数解码。session_encode()函数会对该用户的所有会话变量编码，而不只是对执行 session_encode()的脚本中注册的变量进行编码。session_encode()函数语法格式如下所示：

```
string session_encode()
```

由函数 session_encode()语法知，该函数的返回值为编码后的字符串。下面创建一些会话变量并赋值，然后使用 session_encode()函数编码，并获取编码后的字符串，如代码 6.9 所示。

代码 6.9　使用 session_encode()函数

```
<?php
 session_start();
 $_SESSION["admin"] = "yound";
 $_SESSION["PWD"]= "youndTang";
 $userString = session_encode();
 echo $userString;
?>
```

执行该段代码，结果如下所示：

```
admin|s:5:"yound";PWD|s:9:"youndTang"
```

2．session_decode()解码函数

编码会话函数所得到的字符串可以通过 session_decode()函数解码，该函数将变量解码，返回最初的格式，如果执行成功则返回 true，否则返回 false。该函数的语法格式如下所示：

```
bool session_decode(string $userstring)
```

下面创建一个示例，该示例是由本节上一个示例演化而来，在本示例中首先使用函数 unset 删除会话变量，然后使用 session_decode()函数解码，如代码 6.10 所示。

代码 6.10　使用 session_decode()函数解码

```
<?php
 session_start();
 $_SESSION["admin"] = "yound";
 $_SESSION["PWD"]= "youndTang";
 $userString = session_encode();
 echo "编码数据: ".$userString;
 unset($_SESSION["admin"]);
 unset($_SESSION["PWD"]);
 echo "<hr/>";
 echo "用户名: ".$_SESSION['admin'];
 echo "<br/>";
 echo "密码: ".$_SESSION['PWD'];
 echo "<hr/>";
 session_decode($userString);
 echo "解码后变量数据: <br/>";
 echo "用户名: ".$_SESSION['admin'];
 echo "<br/>";
 echo "密码: ".$_SESSION['PWD'];
?>
```

执行该段代码，结果如下所示：

```
编码数据: admin|s:5:"yound";PWD|s:9:"youndTang";
--------------------------------------------------------------------
用户名:
密码:
--------------------------------------------------------------------
解码后变量数据:
用户名: yound
密码: youndTang
```

6.3　会话实例

通过前面章节对 cookie 和会话的学习，读者应该了解如何创建 cookie 和会话以及会话的应用，本节将综合前面学习的知识点创建两个实例，以巩固前面的学习。

6.3.1　删除会话中已创建变量

（1）打开 Zend Studio - 5.5.0 工具，首先在该工具中创建一个.php 文件，该文件主要是实

现信息输入，在该文件中使用 HTML 表单实现信息输入页面，其主要代码如下所示。

```
<html>
  <head>
   <title>删除会话中已创建变量</title>
  </head>
  <body>
     <form action="Example2.php">
       输入临时名称<input type=text name=name1>
       <br/><input type=submit  value=提交>
      </form>
  </body>
</html>
```

（2）实现会话绑定与使用。当从客户端获取了会话信息后，就可以将获取的信息使用会话绑定，并将会话信息输出。新建一个.php 文件并重名为 Example2.php，在该页面上输入以下代码：

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"/>
<body  background="admin.gif">
<center><img src=title.jpg width=92% ></center>
<?php
session_start();
$name=$_GET['name1'];
$_SESSION['username']=$name;
echo $_SESSION['username']."欢迎登录网站";
?>
<form action="Example3.php" method="POST">
<br/><input type=submit  value='删除临时名称'>
</form>
```

（3）实现会话变量删除。当浏览者退出此网站时，会话会自动结束。如果要手动结束会话，可以将当前注册在会话中的变量删除。新建 Example3.php 文件，并在该页面输入以下代码：

```
<?php
session start();
if(isset($_SESSION['username']))
{
    unset($_SESSION['username']);
    echo "临时名称已被删除<br/>";
    echo "<a href = 'Example1.php'>转到主页</a>";
}
else
```

```
    {
        echo "该会话变量不存在，删除失败 ";
    }
    ?>
```

（4）打开 IE 浏览器，在地址栏中输入 http://localhost/ Example1.php，单击【转到】按钮，会显示如图 6-6 所示页面效果。



图 6-6　信息输入页面效果图

在图 6-6 所示窗口文本框中输入信息后，单击【提交】按钮，会显示如图 6-7 所示页面效果。

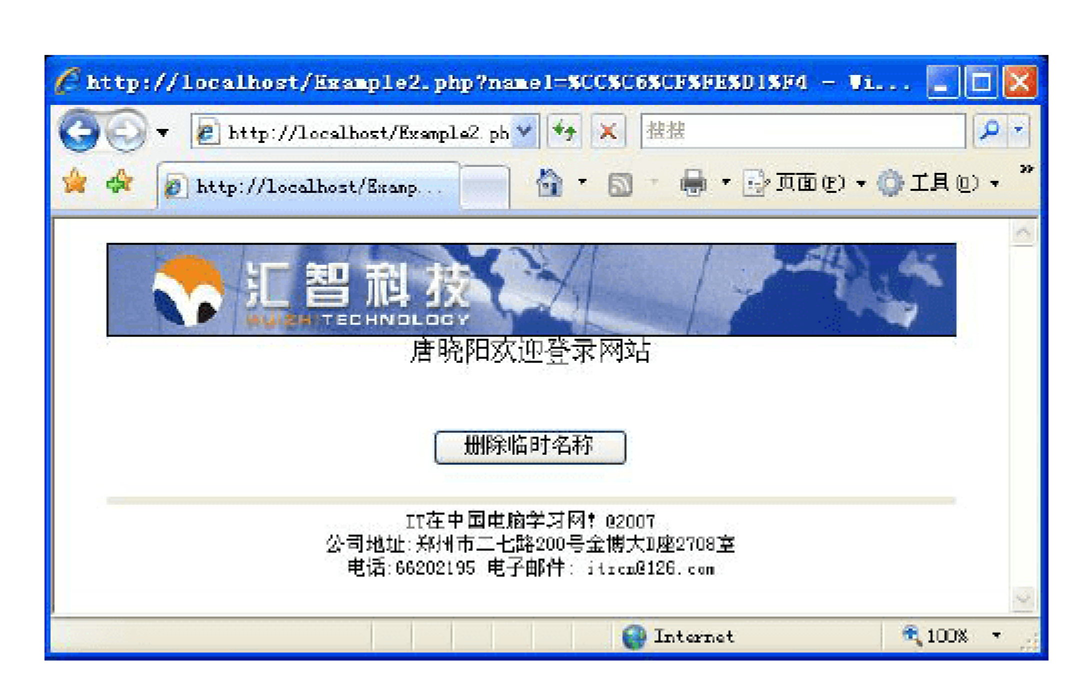


图 6-7　会话信息绑定显示

在图 6-7 所示的页面上，单击【删除临时名称】按钮，删除会话变量，会显示如图 6-8 所示的页面效果。

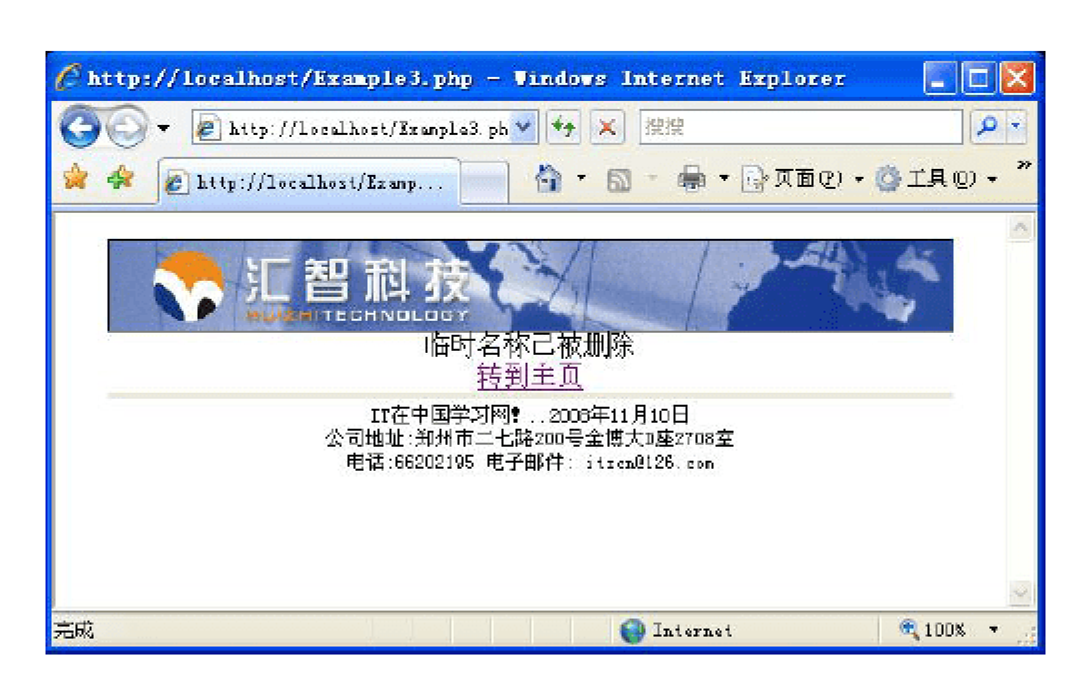


图 6-8　删除会话变量页面

6.3.2　Session 的使用

本节主要是创建一个使用 Session 的示例，通过创建该示例，详细讲解 Session 的具体运用。下面详细介绍该示例的开发步骤。

（1）打开 Zend Studio - 5.5.0，创建 Example-Session.php 和 Example_ Session.php 文件，分别用来实现用户登录和会话绑定等功能。

（2）实现用户登录页面。为了将用户信息和 session 进行关联，首先需要获取用户的相关信息，打开 Example_ Session.php 页面，结合 HTML 表单实现用户登录页面。其代码如下所示。

```
<html>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"/>
<body>
<center><img src=title.jpg></center>
<center>
<table width="500" border="0" cellpadding="0" cellspacing="0" >
<tr><td align="left">用户登录并绑定<hr/></td></tr>
  <tr><td>
    <form action="Example-Session2.php" method="post">
    <table width="100%" border="0" cellpadding="0" cellspacing="0">
      <tr><td align="right">姓名: </td>
         <td><input type="text" size="20" name="username"></td></tr>
      <tr><td align="right">密码: </td>
        <td><input type="password" size="20" name="pwd"></td></tr>
      <tr><td></td><td>
          <input type="submit" value="提交" >
         <input type="reset" value="重置" ></td></tr>
    </table></form>
    </td></tr></table></center>
```

（3）启动 Session 会话。因为在本页面使用了 Session 会话并且调用了 Session 变量，所以必须在调用 Session 之前使用 session_start()函数启动会话。在调用 session_start()之前不能有任何输出，所以将此语句放在页面的首行。当从客户端获取了提交信息之后，就可以利用 session 与信息进行绑定，该步骤主要代码如下所示。

```
<?php
   session_start();
?>
<html>
<body >
<center><img src=title.jpg></center>
<center>
<?php
     if (isset($_POST['username']))
```

```
        {
          $username = $_POST['username'];
          $pwd = $_POST['pwd'];
          if ($username=="唐晓阳" and $pwd=="1225")
          {
            $_SESSION['username'] = $username;
            $_SESSION['password'] = $pwd;
              echo "数据已经顺利存储在 Session 中，用户成功登录！";
              echo "<hr/>";
            echo $username.": 欢迎你访问本站！";
          }
          else
          {
              echo "用户登录失败！";
              echo "<hr/>";
              echo "用户或密码错误，请重新登录！";
            echo "<br/><a href='Example-Session.php'>重新输入用户名</a>";
          }
      }
    ?>
    </center>
```

上述程序运行时，首先使用 isset()函数检测从客户端获取的数据是否存在，如果存在，则使用$_POST 获取客户端提交的数据，并赋值给$username 和$pwd。接下来对提交信息进行校验，如果提交信息通过校验，则使用“$_SESSION['username'] = $username”语句将信息和 Session 进行关联，即通过 Session 名称 username 可以获取$username 变量的值，并输出用户已经成功登录等信息。如果校验不能通过，则显示用户登录失败。

（4）运行。打开 IE 浏览器，在地址栏中输入 http://localhost/Example-Session.php，单击【转到】按钮，会显示如图 6-9 所示窗口。



图 6-9 用户信息输入页面效果图

在图 6-9 中的文本框中输入信息后，单击【提交】按钮，会将信息提交给后台处理程序。如果信息通过验证，则显示如图 6-10 所示窗口，如果信息没有通过验证，则显示如图 6-11 所示窗口。

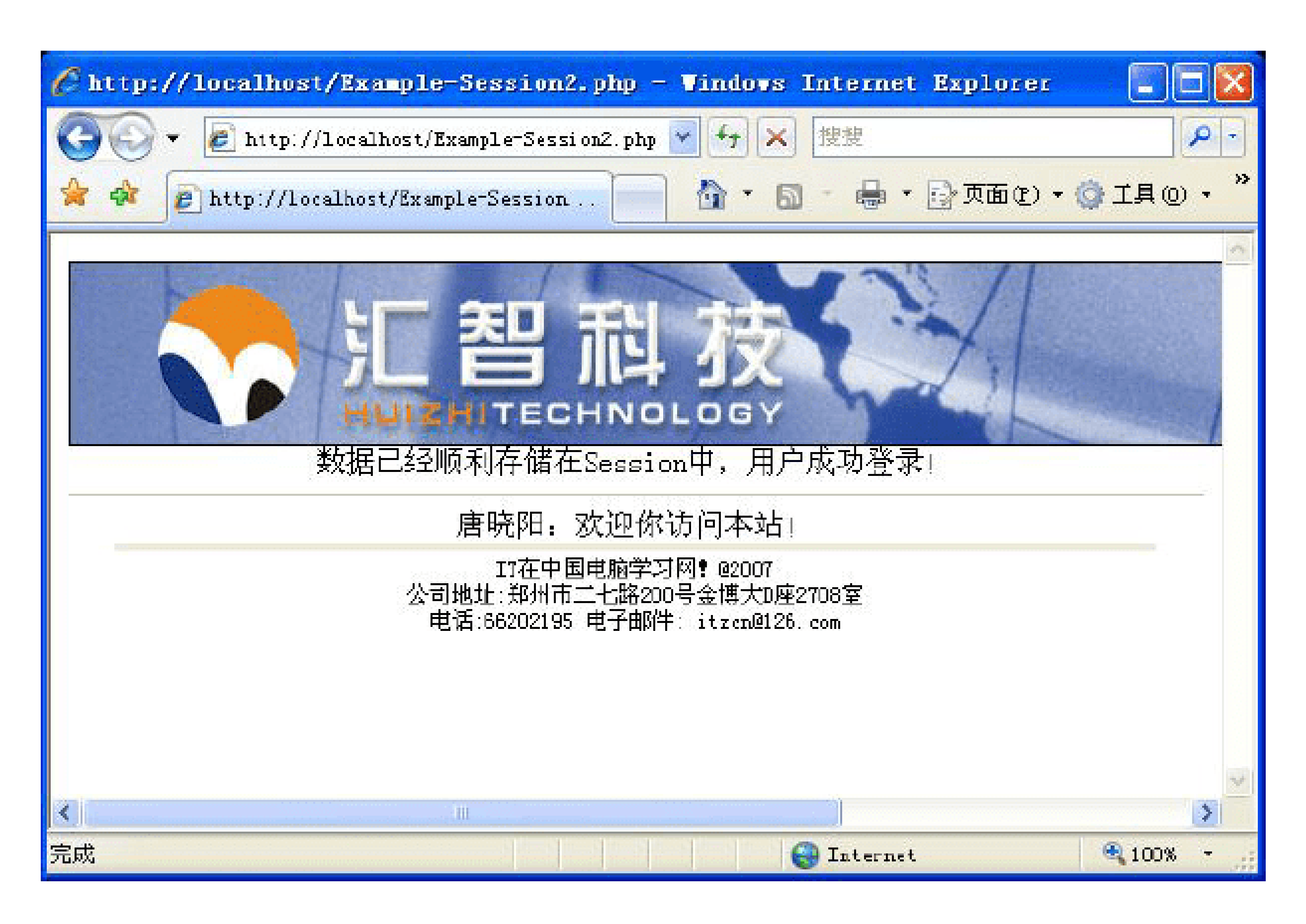


图 6-10　用户成功登录页面效果图



图 6-11　用户登录失败页面效果图

（5）检测。执行完程序后，可以到系统临时文件夹 C:\WINDOWS\Temp 找到这个 Session 文件，一般文件名形如：sess_9ic899ljfcn38sn39nhs28b122，后面是 32 位编码后的随机字符串。使用记事本方式打开该文件，其内容如下所示：

```
username|s:6:"唐晓阳";password|s:4:"1225";
```

第7章 PHP 操作文件和数据库



动态网站技术一个最重要特征就是数据的交互，既可以从客户端向 MySQL 服务器端插入数据，也可以从服务器端获取并显示数据，或者是操作服务器上的文件，例如处理文件系统的输入和输出。完成这样的操作需要使用 PHP 操作文件，或者使用 PHP 连接数据库，从而操作数据库中的数据。本章主要介绍如何操作文件、连接数据库并执行查询语句、准备语句和事务处理等。

学习目标 Objective

- 了解如何解析目录路径
- 掌握访问文件/目录属性的函数
- 灵活运用打开和关闭文件函数
- 熟练读取与写入文件
- 熟练连接 MySQL 数据库
- 掌握如何在 PHP 中执行 SQL 语句
- 掌握获取数据和显示数据的函数
- 熟练管理数据库中的数据
- 了解多个查询、准备语句和事务处理

7.1 文件目录和属性

在 PHP 中，访问系统中的文件是通过文件路径找到文件，然后再获取文件的相关属性信息或者操作文件。通过使用 PHP 内置函数可以解析文件路径、修改文件和获取目录重要信息等。本节主要讲解这些内置函数，以及如何使用这些函数获取文件和目录相关信息，例如位置、大小、修改时间和最后访问时间等。

7.1.1 解析目录路径

在 PHP 中有很多可以把路径解析为各个属性的函数，使用这些函数可以轻松得到路径中所包含的一些属性信息，例如路径末尾文件的扩展名、目录部分和文件名等。本节主要讲解

解析目录路径的函数。

1．获取文件名函数

在 PHP 中可以使用 basename()函数获取路径中文件名部分，该函数语法格式如下：

```
string basename(string path [,string suffix])
```

由该函数语法格式可知，basename 函数有两个参数：参数 path 表示需要检查的路径；参数 suffix 为可选参数，表示文件扩展名。如果在使用该函数时提供了 suffix 参数，并且返回的文件名包含该扩展名，那么输出时将忽略此扩展名。下面使用该函数创建一个示例，如代码 7.1 所示。

代码 7.1　使用 basename()函数

```
<fieldset><legend>使用 basename 函数示例</legend>
<?php
  $path ="/test/home.php";
  $suffix =".php";
  $suffix1 = ".txt";
  echo "<table><tr><td align=\"right\">
       没有可选参数: </td><td>".basename($path)."</td></tr>";
  echo "<tr><td align=\"right\">文件名中包含可选参数扩展名:
       </td><td>".basename($path,$suffix)."</td></tr>";
  echo "<tr><td align=\"right\">文件名中不包含可选参数扩展名:
       </td><td>".basename($path,$suffix1)."</td></tr></table>";
?></fieldset>
```

保存代码 7.1 到“F:\MyWeb\Apache\htdocs\7”目录下，并重名为“Demo-FilePath.php”，打开 IE 浏览器，在地址栏中输入“http://localhost/7/Demo-FilePath.php”，然后单击【转到】按钮打开页面，效果如图 7-1 所示。

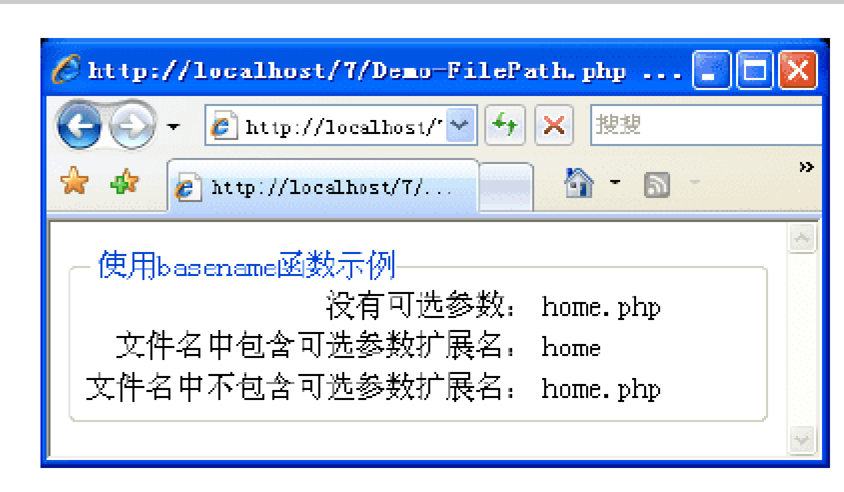


图 7-1　页面效果图

2．获取目录部分函数

函数 dirname()可以得到路径 path 的目录部分，该函数的语法格式如下：

```
string dirname(string path)
```

dirname()函数的 path 参数是指向一个文件的全路径的字符串，该函数返回去掉文件名后的目录名。接下来在上面示例中使用该函数获取目录部分，具体代码如下：

```
  $path ="/test/home.php";
  echo dirname($path);
```

保存代码并执行，结果如下：

```
/test
```

3. 获取路径中各个部分函数

pathinfo()函数以数组的形式返回文件路径信息。该函数可以创建一个关联数组，其中包括 path 指定的路径中的 4 个部分：目录名、基本名、扩展名和文件名，分别通过数组键 dirname、basename、extension 和 filename 来引用，该函数语法格式如下：

```
array pathinfo(string path)
```


pathinfo()函数有一个参数，表示路径字符串。下面创建一个示例，如代码 7.2 所示。

代码 7.2　使用 pathinfo()函数

```
<?php
$path ="web/test/index.php";
$pathinfo = pathinfo($path);
echo "直接输出数组：<br/>";
print r($pathinfo);
echo "<br/>分解数组后输出：<br/>";
echo "目录名：".$pathinfo['dirname']."<br/>";
echo "基本名：".$pathinfo['basename']."<br/>";
echo "扩展名：".$pathinfo['extension']."<br/>";
echo "文件名：".$pathinfo['filename']."<br/>";
?>
```

保存该段代码并执行，结果如下：

```
直接输出数组:
Array
(
   [dirname] => web/test
   [basename] => index.php
   [extension] => php
   [filename] => index
)
分解数组后输出:
目录名：web/test
基本名：index.php
扩展名：php
文件名：index
```

4. 获取绝对路径或硬链接函数

realpath()函数返回绝对路径，语法格式如下：

```
string realpath(string path)
```

该函数将 path 中的所有符号链接和相对路径引用转换为相应的硬链接和绝对路径。例如

在“F:\MyWeb\Apache\htdocs\7\1”目录下有一个文本文件 1.txt，可以使用 realpath()函数解析所有本地路径引用，具体代码如下：

```
<?php
$path = "1/1.txt";
echo realpath($path);
?>
```

保存该段代码，执行结果如下：

```
F:\MyWeb\Apache\htdocs\7\1\1.txt
```

7.1.2　访问文件属性

在 PHP 中有许多访问文件属性的函数，通过使用这些函数可以获得文件属性，例如获取文件类型、大小、最后一次修改时间、最后一次访问时间、文件所属组、文件是否可读和文件是否可写等。本节主要讲解访问文件属性的相关函数。

1. 获取文件类型函数

filetype()函数可以获取指定文件或目录的类型，语法格式如下：

```
filetype(string filename)
```

如果成功执行 filetype()函数，则返回 8 种可能的值，如表 7-1 所示，否则返回 false。

表 7-1　filetype()函数执行成功返回值介绍

返回值名称	说明
block	块设备，例如软件驱动器或 CD-ROM
char	字符设备，负责操作系统和设备之间进行无缓冲的数据交换
dir	目录
fifo	命名管道，常用于将信息从一个进程传递到另一个进程
file	硬链接，作为文件 inode 的指针。只要认为是一个文件，例如文本文档或可执行文件，都返回这个类型
link	符号链接，指向文件指针的指针
socket	套接字资源
unknown	未知类型

下面创建一个示例，具体讲解如何运用 filetype()函数，如代码 7.3 所示。

代码 7.3　运用 filetype()函数获取文件或目录类型

```
<?php
$filename ="filetype.txt";
$filePath ="1";
echo filetype($filename)."<br/>";
```

```
echo filetype($filePath);
?>
```

保存并执行该段代码，结果如下：

```
file
dir
```

2. 获取文件大小函数

在 PHP 中，可以使用 filesize()函数获取指定文件的大小，该函数语法格式如下：

```
filesize(string filename)
```

如果 filesize()函数执行成功，则返回文件大小的字节数，否则返回 false 并生成一条 E_WARNING 级的错误。下面使用该函数创建一个示例，代码如下：

```
<?php
$filename = "1/1.txt";
$size = filesize($filename);
echo "文件".basename($filename).
    "为".$size."字节或者".round($size/1024,2)."KB";
?>
```

保存并执行该段代码，结果如下：

```
文件 1.txt 为 8050 字节或者 7.86KB
```

3. 获取文件 inode 编号函数

fileinode()函数返回文件的 inode 编号，语法格式如下：

```
fileinode(string filename)
```

如果 fileinode()函数执行成功，则返回指定文件 inode 节点号，否则返回 false。下面使用该函数创建一个示例，具体代码如下：

```
<?php
$filename ="1/1.txt";
echo fileinode($filename);
?>
```

保存该段代码，执行结果如下：

```
0
```

4. 返回用户最后一次访问文件时间函数

在网站应用程序中，用户经常需要查看文件最后一次访问时间，这就需要用到 PHP 内置函数 fileatime()。该函数返回指定文件上次访问的时间，语法格式如下：

```
fileatime(string filename)
```

fileatime()函数返回文件上次被访问的时间，如果出错则返回 false。下面创建一个示例，具体讲解如何使用该函数返回文件上次被访问的时间，代码如下：

```
<?php
$filename ="1/1.txt";
echo fileatime($filename) ."<br/>";
echo "文件".basename($filename)."最后一次
     被访问的时间是: ".date("m-d-y g:i:sa",fileatime($filename));
?>
```

保存并执行该段代码，结果如下：

```
01227110400
文件 1.txt 最后一次被访问的时间是: 11-19-08 4:00:00pm
```

5. 获取文件最后一次修改时间函数

在 PHP 中，开发人员可以使用 filectime()函数获取文件最后一次被修改的时间，该函数的语法格式如下：

```
filectime(string filename)
```

该函数如果执行失败则返回 false。下面创建一个示例，具体代码如下：

```
<?php
$filename ="1/1.txt";
echo filectime($filename)."<br/>";
echo "文件".basename($filename)."最后一次
     被修改的时间是: ".date("m-d-y g:i:sa",filectime($filename));
?>
```

保存并执行该段代码，结果如下：

```
1227163328
文件 1.txt 最后一次被修改的时间是: 11-20-08 6:42:08am
```

6. 获取文件上次修改时间函数

filemtime()函数用于返回文件 filename 上次被修改的时间，如果出错则返回 false。时间以 UNIX 时间戳的方式返回。这里的修改时间指的是对文件内容进行修改的时间。该函数语法格式如下：

```
int filemtime ( string filename)
```

下面创建一个示例，具体讲解如何使用该函数获得文件上次被修改的时间，具体代码如下：

```
<?php
$filename = '1/1.txt';
if (file_exists($filename))
{
     echo filemtime($filename)."<br/>";
    echo basename($filename)." 文件上次被修改的时间是:
        " . date ("F d Y H:i:s.", filemtime($filename));
}
?>
```

保存并执行该段代码，结果如下所示：

```
1227168130
1.txt 文件上次被修改的时间是: November 20 2008 08:02:10.
```

7．获取文件相关信息函数

stat()函数返回文件的相关信息，语法格式如下：

```
array stat(string filename)
```

该函数用于获取由 filename 指定的文件的统计信息。如果 filename 是符号链接，则统计信息是关于被链接文件本身，而不是符号链接。stat()函数返回一个数组包含文件的统计信息，该数组具有的单元如表 7-2 所示，并且数组下标从零开始。除了数字索引之外，自 PHP 4.0.6 起还可以通过关联索引来访问。跟该函数类似的函数还有 fstat()函数和 lstat()函数，这 3 个函数的用法类似，并且这 3 个函数返回相同的数组，不同的是 fstat()函数作用于打开的文件指针而不是文件名；stat()函数需要实际的文件名和路径而不是指针；lstat()函数和 stat()函数一样，唯一的不同是如果 filename 参数是符号链接，则该符号链接的状态被返回，而不是该符号链接所指向的文件状态。在此只详细讲解 stat()函数。

表 7-2　返回数组具有的单元

数组下表	关联键名	说明
0	dev	文件所在的设备号
1	ino	文件的 inode 号
2	mod	文件 inode 保护模式，这个值确定指派给文件的访问和修改权限
3	nlink	与该文件关联的硬链接数目
4	uid	文件所有者的用户 ID（UID）
5	gid	文件所有者的组 ID（GID）
6	rdev	设备类型
7	size	文件大小，以字节为单位
8	atime	文件的最后访问时间（UNIX 时间戳格式）
9	mtime	最后修改时间（UNIX 时间戳格式）
10	ctime	最后改变时间（UNIX 时间戳格式）
11	blksize	文件系统的块大小，注意 Windows 平台不可用
12	blocks	所占据块的数目

下面创建一个示例，具体讲解如何使用这 3 个函数获得文件信息，如代码 7.4 所示。

代码 7.4　获取文件信息

```
<fieldset><legend>获取文件信息</legend>
<?php
$filename = "1/1.txt";
//=====打开 filename 文件，fopen()函数将在下面讲解==
$fh = fopen($filename,"r");
$fileinfo = fstat($fh);
echo "fstat 函数输出的文件信息: <br/>";
print r(fstat($fh));
echo "<br/>".$fileinfo['size'];
echo "<hr/>";
echo "stat 函数输出的文件信息: <br/>";
print_r(stat($filename));
echo "<hr/>";
echo "lstat 函数输出的文件信息: <br/>";
print_r(lstat($filename));
//========关闭打开的文件，fclose()将在下面讲解==
fclose($fh);
?>
</fieldset>
```

保存并执行代码 7.4，页面效果如图 7-2 所示。

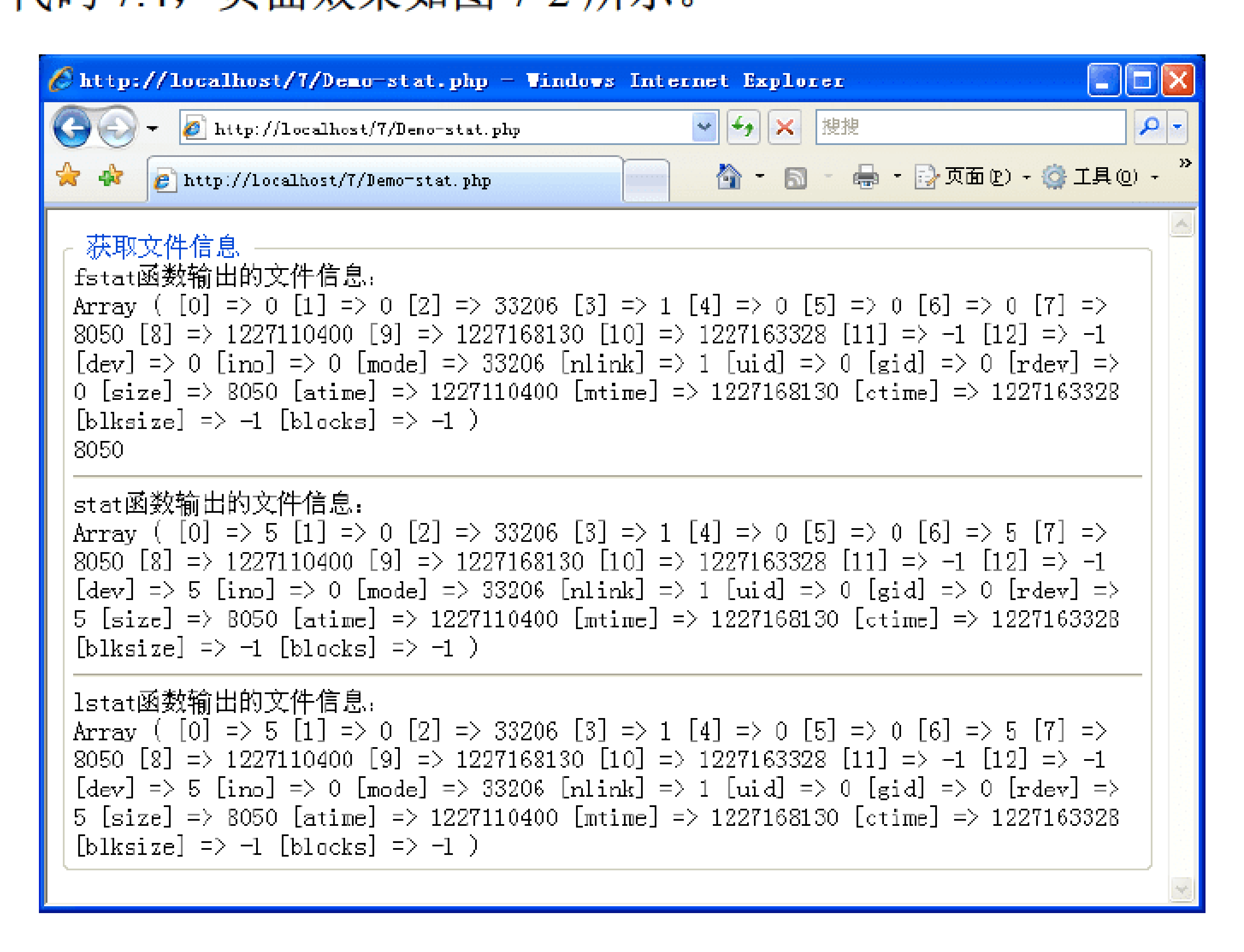


图 7-2　获取文件信息页面执行效果

8．获取文件所属组函数

filegroup()函数以数字格式返回指定文件的组 ID，如果出错则返回 false。该函数语法格式

如下：

```
int filegroup ( string filename)
```

下面创建一个返回组 ID 的示例，具体代码如下：

```
<?php
  $filename = "1/1.txt";
  echo filegroup($filename);
?>
```

9. 获取文件所有者函数

fileowner()函数用于取得文件的所有者，语法格式如下：

```
int fileowner ( string filename)
```

fileowner()函数返回文件所有者的用户 ID，如果出错则返回 false。用户 ID 以数字格式返回，用 posix_getpwuid()来将其解析为用户名。下面创建一个示例，具体介绍如何运用 fileowner()函数，具体代码如下：

```
<?php
  $filename ="1/1.txt";
  echo fileowner($filename);
?>
```

保存并执行该段代码，结果如下：

```
0
```

10. 获取文件或目录权限

fileperms()函数用于返回文件的访问权限，如果出错则返回 false，语法格式如下：

```
int fileperms ( string filename)
```

下面创建一个示例，具体讲解如何使用该函数返回文件权限，以及目录权限，如代码 7.5 所示。

代码 7.5　使用 fileperms()函数返回文件和目录权限

```
<html>
<head><title>fileperms 函数示例</title></head>
<body style="margin-top:0">
<fieldset>
   <img src="top.jpg" width="1000px">
</fieldset>
<fieldset><legend>文件权限示例</legend>
<?php
```

```
    $filename = "1/1.txt";
    echo "<hr/>文件权限：<br/>";
    echo "<font color='red'>".fileperms($filename)."</font>";
    echo "<br/>以 8 进制返回文件权限：<br/>";
    echo "<font color='red'>".substr(sprintf("%o",fileperms($filename)),
    -4)."</font>";
    $filepath ="1";
    echo "<br/>目录权限：<br/>";
    echo "<font color='red'>".fileperms($filepath)."</font>";
?>
</fieldset>
</body>
</html>
```

保存该段代码到“F:\MyWeb\Apache\htdocs\7”目录下，并重命名为“Demo-Group.php”，然后打开 IE 浏览器，在地址栏中输入 http://localhost/7/Demo-Group.php，单击【转到】按钮，页面效果如图 7-3 所示。



图 7-3　页面效果图

11．获取文件是否可执行函数

is_executable()函数用于判断所给文件 filename 是否可执行，如果文件存在且可执行则返回 true，语法格式如下：

```
bool is_executable ( string filename)
```

is_executable()从 PHP 5.0.0 版本起可用于 Windows 系统。下面创建一个示例，判断文件是否可执行，如代码 7.6 所示。

代码 7.6　使用 is_executable()函数判断文件是否可执行

```
<?php
   $filename = "1/1.txt";
   $file="1/setup.exe";
   if(is_executable($filename))
   {
      echo basename($filename)."文件是可执行文件<br/>";
   }
   else
   {
      echo basename($filename)."文件不是可执行文件<br/>";
   }
   if(is_executable($file))
   {
```

```
        echo basename($file)."文件是可执行文件<br/>";
    }
    else
    {
        echo basename($file)."文件不是可执行文件<br/>";
    }
?>
```


保存并执行代码 7.6，结果如下：

```
1.txt 文件不是可执行文件
setup.exe 文件是可执行文件
```

12．判断文件是否可读和可写函数

is_readable()函数用于判断所给文件是否可读。如果由路径指定的文件或目录存在并且可读则返回 true。该函数的语法格式如下：

```
bool is_readable ( string filename)
```

判断文件是否可写的函数是 is_writable()，如果由路径指定的文件或目录存在并且可写则返回 true。该函数语法格式如下：

```
bool is_writable ( string filename)
```

下面创建一个示例，具体讲解如何运用这两个函数，如代码 7.7 所示。

代码 7.7　判断文件是否可读和可写

```
<?php
    $filename = "1/1.txt";
    if(is_readable($filename))
    {
        echo basename($filename)."文件可以读取<br/>";
    }
    else
    {
        echo basename($filename)."文件是不可读取文件<br/>";
    }
    if(is_writable($filename))
    {
        echo basename($filename)."文件可以写入<br/>";
    }
    else
    {
        echo basename($filename)."文件不可以写入<br/>";
    }
?>
```

保存并执行代码 7.7，结果如下：

```
1.txt 文件可以读取
1.txt 文件可以写入
```


7.1.3　访问目录属性信息

在应用程序中，经常需要计算服务器上磁盘空间的大小和磁盘空间的剩余容量，这些功能的实现需要用到 PHP 中的两个函数：disk_free_space()函数和 disk_total_space()函数。本节主要讲解这两个函数和如何在 PHP 中计算目录的大小。

1．获取目录所在磁盘可用空间函数

disk_free_space()函数返回目录所在磁盘分区的可用空间，返回结果以字节为单位，该函数的语法格式如下：

```
float disk_free_space(string directory)
```

下面创建一个示例，具体讲解如何运用 disk_free_space()函数获取目录所在磁盘分区的可用空间，代码如下：

```
<?php
  $drive ="F:\\MyWeb\\Apache\\htdocs\\7\\";
  echo disk_free_space( $drive);
?>
```

保存并执行该段代码，结果如下：

```
14477869056
```

2．获取磁盘总容量函数

disk_total_space()函数用于返回一个目录的磁盘总大小。确切地说返回的是该目录所在磁盘分区的总大小，因此用同一个磁盘分区的不同目录作为参数所得到的结果完全相同。在 UNIX 和 Windows NT 中都支持将一个磁盘分区加载为一个子目录，这时该函数与 disk_free_space()函数结合就很有用。该函数语法格式如下：

```
float disk_total_space ( string directory)
```

下面使用该函数创建一个示例，具体代码如下：

```
<?php
//返回目录所在磁盘的总大小
$df = disk total space("/");
$dfs = round($df/1048576,2);
echo 'dfs='.$dfs.'MB';
```

```
?>
```

上述代码成功执行后，将输出如下信息：

```
dfs=14307.86MB
```


在该示例中，返回值的数量单位是兆字节（MB），这是由于在代码中将 disk_total_space()函数返回值除以 1048576 字节（1MB）而得到的。

3. 获取目录大小

PHP 虽然提供了 filesize()、disk_free_space()及 disk_total_space()函数，但并没有提供获取目录总大小的标准函数。也许会有读者说，可以使用 exec()或者 system()做系统级调用 du 命令（用于获得一个文件或目录的磁盘使用情况），但由于存在安全方面的问题，这些函数通常是禁用的。为此不得不采用另一种解决方法，该方法通过一个自定义的 PHP 函数来完成这个任务。下面创建一个获取目录大小的示例，如代码 7.8 所示。

代码 7.8　获取目录大小

```
<?php
    function get_dirsize($directory) {
      $dirSize=0;
      //打开目录并且读它的内容
      if ($dh = @opendir($directory))
      {
         //读取目录中的每个文件
         while ($filename = readdir ($dh))
         {
            //过滤掉某些未知的文件。
            if ($filename != "." && $filename != "..")
            {
               // 确定文件大小并进行合计
               if (is_file($directory."/".$filename))
                  $dirSize += filesize($directory."/".$filename);
                  if (is_dir($directory."/".$filename))
                     $dirSize += directory_size($directory."/".$filename);
            }
         }
      }
     @closedir($dh);
     return $dirSize;
   }
   $mydir = "1/";
   $totalSize = round((get_dirsize($mydir) / 1024), 2);
   echo "目录".$mydir."大小为: ".$totalSize. "kB.";
?>
```

保存并执行代码 7.8，结果如下：

```
目录 1/大小为：418.17kB.
```

7.2　操作文件

程序开发人员经常访问目录和文件，以便收集信息和对文件系统进行必要的更改，例如访问文件系统、读取或写入文件、移动或复制文件或浏览文件夹等。本节主要讲解如何在 PHP 中实现这些功能，以及实现这些功能所涉及到的函数。

7.2.1　打开和关闭文件

在应用程序中，如果想操作系统中的文件，首先要打开文件，然后才能进行读取、写入和复制粘贴等操作，最后再关闭文件。本节主要讲解在 PHP 中打开和关闭文件的两个标准函数。

1．打开文件函数

fopen()函数可以打开文件和 URL，如果该执行失败则返回 false。该函数的语法格式如下：

```
resource fopen(string filename ,string mode [,int use_include-path , string context])
```

fopen()函数用于将 filename 指定的文件资源绑定到一个流或句柄上，绑定之后，脚本就可以通过句柄与资源进行交互。如果 filename 是"scheme://..."的格式，则被当成一个 URL，PHP 将以搜索协议处理器来处理此模式。如果该协议尚未注册封装协议，PHP 将发出一条消息来帮助检查脚本中潜在的问题，并将 filename 当成一个普通的文件名继续执行下去。

如果 PHP 认为 filename 指定的是一个本地文件，将尝试在该文件上打开一个流。该文件必须是 PHP 可以访问的，因此需要确认文件访问权限允许访问。如果 PHP 认为 filename 指定的是一个已注册的协议，而该协议被注册为一个网络 URL，PHP 将检查并确认 allow_url_fopen 已被激活。如果关闭了，PHP 将发出一个警告，而 fopen 的调用则失败。

fopen()函数中 mode 参数表示的是访问文件/流的类型，主要是确定用户访问资源的级别，该参数可能的值如表 7-3 所示。

表 7-3　mode 参数可以指定的模式

mode	说明
"r"	只读方式打开，将文件指针指向文件头
"r+"	读写方式打开，将文件指针指向文件头
"w"	写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之
"w+"	读写方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之
"a"	写入方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建一个新文件

续表

mode	说明
"a+"	读写方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建一个新文件
"x"	创建并以写入方式打开，将文件指针指向文件头。如果文件已存在，则 fopen() 调用失败返回 false，并生成一条 E_WARNING 级别的错误信息。如果文件不存在则尝试创建之，这和给底层的 open(2)系统调用指定 O_EXCL|O_CREAT 标记是等价的。此选项被 PHP 4.3.2 以及以后的版本所支持，仅能用于本地文件
"x+"	创建并以读写方式打开，将文件指针指向文件头。如果文件已存在，则 fopen() 调用失败返回 false，并生成一条 E_WARNING 级别的错误信息。如果文件不存在则尝试创建之。这和给底层的 open(2)系统调用指定 O_EXCL|O_CREAT 标记是等价的。此选项被 PHP 4.3.2 以及以后的版本所支持，仅能用于本地文件

如果资源是位于本地文件，PHP 则认为可以使用本地路径或相对路径来访问此资源，或者可以将 fopen()的 use_include_path 参数设置为 1，这样会使 PHP 考虑配置指令 include_path 中指定的路径。参数 content 用来设置文件或流特有的配置参数，以及在多个 fopen()请求之间共享文件或者流特有的信息。下面创建一个打开文件的具体示例，如代码 7.9 所示。

代码 7.9 打开文件

```
<?php
$file = fopen("filetype.txt","r");
$file = fopen ("1/1.txt", "r");
$file = fopen ("1/1.txt","w");
$file = fopen ("http://www.itzcn.com/", "r");
$file = fopen ("ftp://user:password@ example.com/text.txt", "w");
?>
```

2. 关闭文件函数

在 PHP 中，可以使用 fclose()函数关闭已经打开的文件，语法格式如下：

```
bool fclose ( resource handle)
```

fclose()函数将关闭之前打开的由 handle 指定的文件指针，如果成功则返回 true，否则返回 false。参数 handle 必须是 fopen()或者 fsockopen()成功打开的文件指针。下面创建一个使用该函数的示例，具体代码如下：

```
<?php
  $handle = fopen('filetype.txt', 'r');
  fclose($handle);
?>
```

7.2.2 读取文件

使用 fopen()函数打开文件以后，就可以使用 PHP 内置的读取文件的函数读取文件中的数

据，这些函数不仅可以一次只读取一个字符，而且还可以一次读取整个文件。本节将介绍一些 PHP 中读取文件的内置函数。

1. file()函数

在 PHP 中使用 file()函数可以将整个文件读入到一个数组中，各元素由换行符分隔，同时换行符仍附加在每个元素的末尾，该函数的语法格式如下：

```
array file ( string filename [, int use_include_path [, resource context]])
```

file()函数如果读取文件失败则返回 false，如果想在 include_path 中搜寻文件的话，可以将可选参数 use_include_path 设为 1。下面使用该函数创建一个示例，具体讲解在 PHP 中如何读取系统中的文件。在创建示例之前，首先创建一个文本文档，并在文本文档中存入数据，内容如下：

```
读取文件示例:
北京·北京 天津·天津
上海·上海 重庆·重庆
河南·郑州 湖北·武汉
广东·广州 四川·成都
陕西·西安 浙江·杭州
```

下面编写该示例的 PHP 脚本程序，以读取 FileInfo.txt 文件中的数据，具体如代码 7.10 所示。

代码 7.10　读取文件中内容

```
<?php
//===使用 file 函数将文件内容读入数组中==
$lines = file('FileInfo.txt');
// 在数组中循环，显示 FileInfo.txt 文件内容
$i = 0;
foreach ($lines as  $line)
{
    if($i==0)
    {
      $i++;
      echo $line."<br/>";
    }
    else
    {
      list($sheng1,$sheng2) = explode(' ',$line);
     echo "省份·省会 : ".$sheng1."<br/>";
     echo "省份·省会 : ".$sheng2."<br/>";
    }
}
?>
```

保存并执行代码 7.10，页面输出结果如下：

```
读取文件示例:
省份·省会 : 北京·北京
省份·省会 : 天津·天津
省份·省会 : 上海·上海
省份·省会 : 重庆·重庆
省份·省会 : 河南·郑州
省份·省会 : 湖北·武汉
省份·省会 : 广东·广州
省份·省会 : 四川·成都
省份·省会 : 陕西·西安
省份·省会 : 浙江·杭州
```

2. file_get_contents()函数

file_get_contents()函数用于将整个文件读入一个字符串中，其使用方法和 file()一样，不同的是 file_get_contents()函数将文件读入一个字符串，该函数语法格式如下：

```
string file_get_contents(string filename [, bool use_include_path [, resource
context [, int offset [, int maxlength]]]] );
```

file_get_contents()函数将文件的内容读入到一个字符串中，如果操作系统支持，还会使用内存映射技术来增强性能。由该函数语法格式知，该函数有 5 个参数：offset 参数表示读取文件时开始读取的位置，maxlength 表示此次需要读取的字节数；context 是一套可以修改流的行为的选项；offset 规定在文件中开始读取的位；maxlength 规定读取的字节数。下面使用该函数创建一个示例，具体代码如下：

```
<?php
print r(file get contents("FileInfo.txt"));
?>
```

保存并执行该段代码，结果如下：

```
读取文件示例: 北京·北京 天津·天津上海·上海 重庆·重庆河南·郑州 湖北·武汉广东·广州 四川·成
都陕西·西安 浙江·杭州
```

3. fread()函数

fread()函数可以读取已经打开的文件，并且可以规定读取的字符数。该函数的语法格式如下：

```
string fread(resource handle,int length)
```

该函数从 handle 指定的资源中读取 length 个字符，当到达 EOF 时、读取到 length 个字符时、当一个包可用时和已读取了 8192 个字符时就会停止读取文件，如果出错则返回 false。



fread()函数与其他读取函数不同，使用该函数时，不考虑换行符；因此只要使用 filesize()函数确定了文件的字符数，就能很方便地读取整个文件。

下面创建一个示例，具体讲解如何运用该函数，代码如下：

```
<?php
$filename = "FileInfo.txt";
$file = fopen($filename,"r");
$getFile = fread($file,filesize($filename));
echo $getFile;
fclose($file);
?>
```

保存并执行该段代码，结果和上一个示例相同，在此就不再提供执行结果。

4．fgetc()函数

fgetc()函数从已经打开的文件指针中读取字符，并且只返回一个字符，该函数的语法格式如下：

```
string fgetc( resource $handle)
```

该函数返回的字符从 handle 指向的文件中得到，如果遇到 EOF 则返回 false。这里需要注意，文件指针必须有效，并且必须指向一个由 fopen()或 fsockopen()成功打开的文件。下面使用该函数创建一个示例，具体代码如下：

```
<?php
$file = fopen("test.txt","r");
echo fgetc($file);
fclose($file);
?>
```

5．fgets()函数

fgets()函数从打开文件的指针中读取一行并返回长度最多为 length–1 字节的字符串。该函数语法格式如下：

```
string fgets(int handle [, int length])
```

该函数在读取数据时遇到换行符（包括在返回值中）、EOF 或者已经读取了 length–1 字节后停止。如果没有指定 length，则默认为 1KB（1024 字节）。如果出错则返回 false。下面使用该函数创建一个示例，具体代码如下：

```
<?php
$fp = fopen("FileInfo.txt", "r");
echo fgets($fp);
```

```
echo "<br/>循环读取文件内容<br/><br/>";
while (!feof($fp))
{
  echo fgets($fp)."<br/>";
}
fclose($fp);
?>
```


6. fgetss()函数

该函数与 fgets()函数功能基本相同，只是该函数在读取打开文件时，会自动过滤 HTML 和 PHP 标记，语法格式如下：

```
fgetss(file,length,tags)
```

下面创建一个使用 fgetss()函数的示例，该示例使用 Test.html 文件，文件内容如下：

```
<html><head><title>使用 fgetss 函数</title></head>
<body>使用 fgetss 函数</body></html>
```

在该示例中首先使用 fopen()函数打开文件 Test.html，然后再使用 fgetss()函数读取 Test.html 文件中的内容，具体代码如下：

```
$file = fopen("Test.html","r");
echo fgetss($file);
fclose($file);
```

7. fgetcsv()函数

fgetcsv()函数从文件指针中读取一行，并且解析 CSV 字段，然后再返回一个包含这些字段的数组。语法格式如下：

```
array fgetcsv(int handle [, int length [, string delimiter [, string
enclosure]]]);
```

在使用该函数时，如果出错则返回 false。下面创建一个使用该函数的示例，使用的 test.txt 文件内容如下所示，该示例具体如代码 7.11 所示。

```
河南·郑州;
湖北·武汉;
广东·广州;
四川·成都;
陕西·西安;
浙江·杭州;
```

代码 7.11　使用 fgetcsv()函数读取数据

```
<?php
```

```
$row = 1;
$fp = fopen("test.txt","r");
while ($stra = fgetcsv($fp, 1024, ";"))
{
    $num = count($stra);
    echo "第".$row."行数据: ";
    $row++;
    for ($i=0; $i < $num; $i++)
    {
        echo $stra[$i]."<br/>";
    }
}
fclose($fp);
?>
```

上述代码成功执行后，输出如下所示的信息：

```
第 1 行数据: 河南·郑州
第 2 行数据: 湖北·武汉
第 3 行数据: 广东·广州
第 4 行数据: 四川·成都
第 5 行数据: 陕西·西安
第 6 行数据: 浙江·杭州
```

8. fpassthru()函数

fpassthru()函数功能与 fgetcsv()函数功能类似，不同的是 fpassthru()解析读入的行并找出 CSV 格式的字段，然后返回一个包含这些字段的数组。由于这两个函数很相似，在此就不再对 fpassthru()函数进行详细介绍。

9. readfile()函数

该函数读入一个文件并写入到输出缓冲，如果该函数执行成功，则返回从文件中读入的字节数。如果失败，则返回 false。并且在运用中可以通过@readfile()形式调用该函数来隐藏错误信息。语法格式如下：

```
int readfile ( string $filename [, bool $use_include_path [, resource $context]] )
```

下面创建一个使用 readfile()函数的示例，具体代码如下：

```
<?php
$bc = readfile('test.txt');
echo $bc;
?>
```

上述代码成功执行后，输出结果如下：

```
河南·郑州; 湖北·武汉; 广东·广州; 四川·成都; 陕西·西安; 浙江·杭州;76
```

7.2.3 移动文件指针

在读取和写入文件时，需要在文件中跳转，即从不同的位置读取以及写入不同的位置。为了实现上述功能，PHP 提供了 3 个函数，本节主要讲解这 3 个函数。

1. fseek()函数

fseek()函数在打开的文件中定位，将文件指针从当前位置向前或向后移动到新的位置。新位置从文件头开始以字节数度量，成功则返回 0，否则返回–1。该函数的语法格式如下：

```
int fseek(resource handle,int offset [,int whence])
```

该函数将 handle 的指针移动到 offset 指定的位置，如果忽略可选参数 whence，则位置将设置为从文件开始到 offset 字节处，否则 whence 可以设置 3 个可能的值，它将影响指针的位置，如下所示为这 3 个可能的值。

- **SEEK_CUR** 设置指针位置为当前位置加上 offset 字节。
- **SEEK_END** 设置指针为 EOF 加上 offset 字节，在这里，offset 必须设置为负值。
- **SEEK_SET** 设置指针为 offset 字节处，这与忽略 whence 效果相同。

2. ftell()函数

ftell()函数主要是获取打开文件指针的当前位置，语法格式如下：

```
int ftell(resource handle)
```

下面创建一个示例，具体讲解如何运用该函数，代码如下：

```
<?php
$file = fopen("test.txt","r");
echo "文件指针当前位置: ".ftell($file);
fseek($file,"15");
echo "<br/>";
echo "文件指针当前位置: ".ftell($file);
fclose($file);
?>
```

保存该段代码，执行结果如下：

```
文件指针当前位置: 0
文件指针当前位置: 15
```

3. rewind()函数

rewind()函数将文件指针倒回到打开文件的开头，语法格式如下：

```
bool rewind(resource handle)
```

该函数如果执行成功则返回 true，否则返回 false。下面创建一个示例，具体讲解如何运用该函数，如代码 7.12 所示。

代码 7.12　使用 rewind()函数

```
<?php
$file = fopen("test.txt","r");
fseek($file,"15");
if(rewind($file))
{
    echo "文件指针已经倒回到了文件的开头";
}
else
{
    echo "文件执行失败";
}
fclose($file);
?>
```

保存并执行该段代码，结果如下：

```
文件指针已经倒回到了文件的开头
```

7.2.4　写入文件

在 7.2.2 节和 7.2.3 节中分别介绍了如何读取打开的文件内容和移动文件指针，本节将介绍如何在打开的文件中写入内容。为实现该功能，PHP 提供了两个打开写入文件的函数。

1. fwrite()函数

fwrite()函数主要是写入文件，该函数返回写入内容的字节数，语法格式如下：

```
int fwrite(resource handle, string string [, int length])
```

fwrite()将 string 的内容写入文件指针 handle 处，如果指定了 length，当写入了 length 个字节或者写完了 string 以后，写入就会停止。下面创建一个使用该函数写入文件的示例，具体代码如下：

```
<?php
$str ="写入文件的内容";
$file = fopen("test.txt","at");
echo "写入文件的字节数: ".fwrite($file,$str)."<br/>";
echo "写入文件的字节数: ".fwrite($file,$str,10);
?>
```

保存并执行该段代码，结果如下：

```
写入文件的字节数: 14
```

```
写入文件的字节数：10
```

2．fputs()函数

该函数的功能和 fwrite()函数一样，并且用法也基本相同，语法格式如下：

```
int fputs(resource handle, string string [, int length])
```

由于该函数的用法和功能和 fwrite()相同，在此就不再提供使用该函数的示例。

7.2.5　读取目录内容

前面几节详细介绍了打开文件、读取文件和写入文件的相关函数。实际上，读取目录内容的过程与读取文件非常相似，本节将介绍读取目录内容的相关函数。

1．打开与关闭目录函数

opendir()函数打开一个目录的句柄，可用于之后的 closedir()、readdir()和 rewinddir()调用中。该函数的语法格式如下：

```
resource opendir(string path [,resource $context])
```

在 PHP 程序中执行该函数，如果执行成功则返回一个目录流，否则返回 false 以及一个错误，可以通过在函数名前加“@”来隐藏错误的输出。

打开目录以后就需要关闭目录，在 PHP 中可以使用 closedir()函数关闭打开的目录。该函数的语法格式如下：

```
void closedir(resource $dir_handle)
```

下面使用这两个函数创建一个示例，具体代码如下：

```
<?php
$dir="./";
if(is_dir($dir))
{
  $d= opendir($dir);
  closedir($d);
}
?>
```

2．readdir()函数

readdir()函数返回由 opendir()打开的目录句柄中的条目。在 PHP 中执行该函数，如果执行成功，则返回一个文件名，否则返回 false。该函数语法格式如下：

```
string readdir(resource dir_handle)
```

下面使用该函数和本节前面讲解的两个函数创建一个示例，如代码 7.13 所示。

代码 7.13　打开目录读取目录内容

```
<fieldset>
  <legend>打开目录读取目录内容</legend>
<?php
$dir = "./";
if (is_dir($dir))
{
    if ($dh = opendir($dir))
     {
        echo "文件夹类型如下所示: <br/>";
       while (($file = readdir($dh)) !=false)
        {
             if(filetype($dir . $file)=="dir")
              {
                echo "<font color='red'>文件/目录名称: ".$file."——文件类型: " .
                filetype($dir . $file) . "</font><br/>";
              }
              else
              {
                  echo "文件/目录名称: ".$file."——文件类型: " . filetype($dir .
                  $file) . "<br/>";
              }
        }
       closedir($dh);
    }
}
?>
</fieldset>
```

保存并执行该段代码，页面效果如图 7-4 所示。

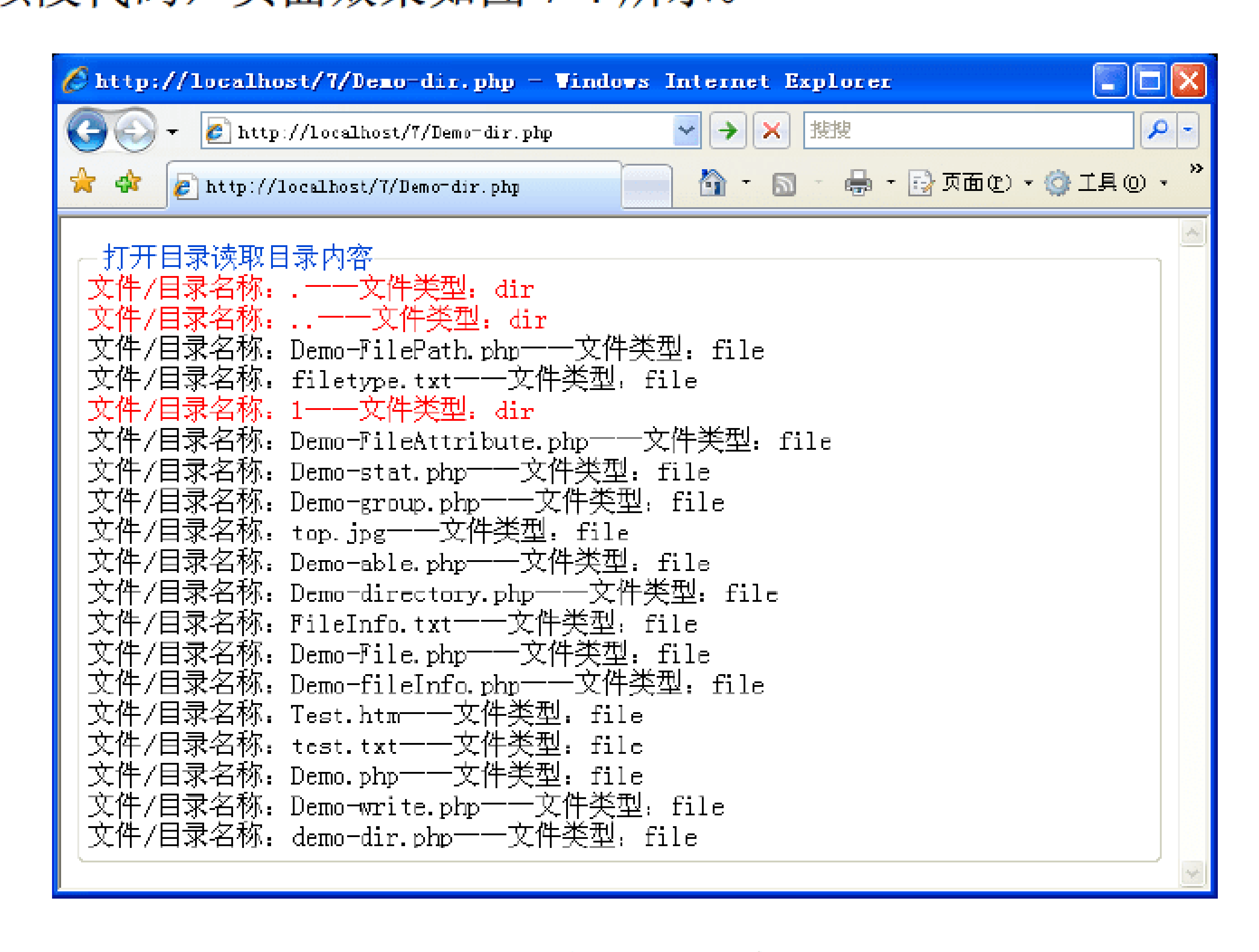


图 7-4　执行页面效果图

3. scandir()函数

scandir()函数返回一个数组，数组中包含指定路径中的文件和目录。在 PHP 中执行该函数，如果执行成功，返回一个数组，否则返回 false。如果 directory 不是目录，则返回布尔值 false，并生成一条 E_WARNING 级的错误。该函数的语法格式如下：


```
array scandir(string directory [,int sorting_order [,resource context]])
```

下面创建一个示例，具体介绍如何运用 scandir()函数，如代码 7.14 所示。

代码 7.14　运用 scandir()函数

```
<fieldset>
  <legend>打开目录读取目录内容</legend>
<?php
$dir ="./";
$files1 = scandir($dir);
$files2 = scandir($dir, 1);
print_r($files1);
echo "<hr/>";
print_r($files2);
?>
</fieldset>
```

保存并执行该段代码，页面效果如图 7-5 所示。

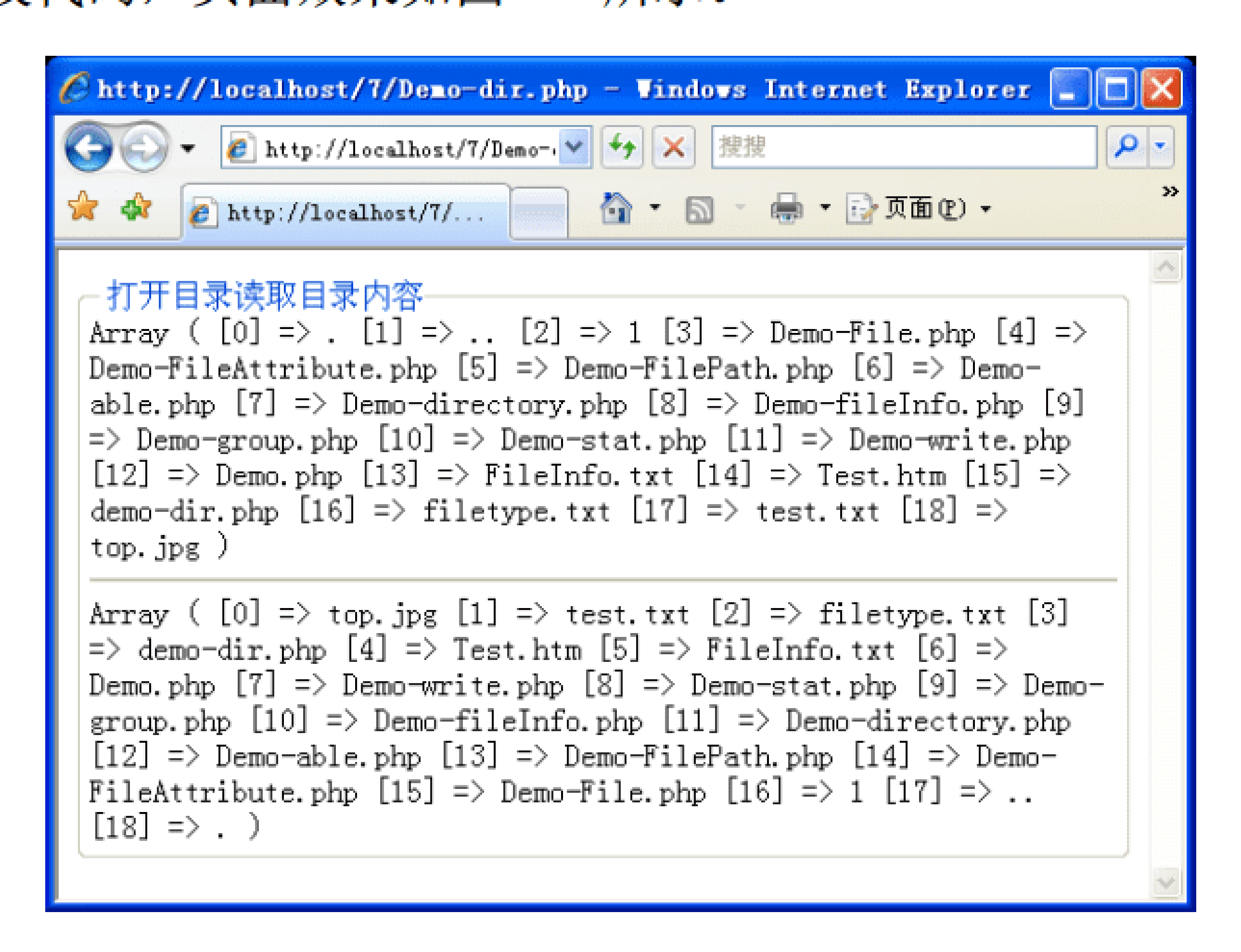


图 7-5　页面效果图

7.3　连接 MySQL 数据库

PHP 最大的一个特点就是能够和数据库紧密结合。多数 PHP 的应用都通过数据库实现，

这就需要 PHP 程序与数据库连接，并操作数据库中的数据。为了轻松地实现这些操作，PHP 提供了大量用于数据库处理的函数。本节主要讲解与数据库连接、关闭连接和选择数据库相关的函数。

7.3.1　建立连接

要进行数据库操作，首先需要连接数据库，也就是 PHP 客户端向服务器端的数据库发出连接请求，连接成功后就可以进行其他的数据库操作。如果使用不同的用户连接，所拥有的操作权限也不同。PHP 连接数据库可以有两种方式，一种是通过 PHP 的 MySQL 相关函数，另一种是通过 PHP 的 ODBC 相关函数。本节主要介绍如何使用 PHP 的 MySQL 函数连接数据库。

1. 打开数据库连接函数

mysql_connect()函数的主要作用是打开或重复使用一个到 MySQL 服务器的非持久性连接，如果要创建持久性连接需要使用 mysql_pconnect()函数，mysql_pconnect()函数将在下面介绍。mysql_connect()函数的语法格式如下：

```
resource mysql_connect ( [string $server [, string $username [, string $password
[, bool $new_link [, int $client_flags]]]]] )
```

由该函数语法格式可知，mysql_connect()函数共有 5 个可选参数，表 7-4 详细介绍了这些参数。

表 7-4　mysql_connect()函数参数说明

参数	描述
server	规定要连接的服务器。可以包括端口号，例如“hostname:port”，或者到本地套接字的路径，例如对于 localhost 的“/path/to/socket”
username	访问 MySQL 数据库要具有的用户名，默认值是服务器进程所有者 root 的用户名
password	用户具有的密码，默认值是空密码
new_link	如果用同样的参数第二次调用 mysql_connect()，将不会建立新连接，而将返回已经打开的连接标识。参数 new_link 改变此行为并使 mysql_connect()总是打开新的连接，即使 mysql_connect()曾在前面被用同样的参数调用过
client_flags	client_flags 参数可以是以下常量的组合。 □ **MYSQL_CLIENT_COMPRESS**　使用压缩协议; □ **MYSQL_CLIENT_IGNORE_SPACE**　允许在函数名后留空格位; □ **MYSQL_CLIENT_INTERACTIVE**　允许设置断开连接之前所空闲等候的 interactive_timeout 时间; □ **MYSQL_CLIENT_SSL**　使用 SSL 加密。本标志仅在 MySQL 客户端库版本为 4.x 或更高版本时可用。在 PHP 4 和 Windows 版的 PHP 5 安装包中绑定的都是 3.23.x

下面创建一个示例，具体讲解如何运用 mysql_connect()函数与数据库建立连接，如代码 7.15 所示。

代码 7.15　与数据库建立连接

```
<?php
  $link=mysql_connect("localhost","root","0");
  if($link)
        {
             echo   "PHP 程序与数据库连接成功";
        }
  else
        {
           echo   "PHP 程序与数据库连接失败";
        }
?>
```

保存并执行代码 7.15，结果如下：

```
PHP 程序与数据库连接成功
```

2. 建立持久性连接函数

mysql_pconnect()函数打开一个到数据库的持久性连接，语法格式如下：

```
resource mysql_pconnect ( [string $server [, string $username [, string $password
[, int $client_flags]]]] )
```

mysql_pconnect()函数与 mysql_connect()函数非常相似，但有两个主要区别。

- 当连接时，myspl_pconnectl 函数将先尝试寻找一个在同一个主机上用同样的用户名和密码已经打开的连接，如果找到，则返回此连接标识而不打开新连接。
- 当脚本执行完毕后到 SQL 服务器的连接不会被关闭，此连接将保持打开以备以后使用，即使使用 mysql_close()函数关闭连接，也不会关闭由 mysql_pconnect()建立的连接。

下面使用该函数创建一个打开连接的示例，代码如下：

```
$link=mysql_pconnect("localhost","root","0");
  if(!$link)
        {
             echo   "数据库连接失败";
        }
  else
        {
           echo   "数据库连接成功";
        }
```

保存并执行该段代码，结果如下：

```
数据库连接成功
```

3. 关闭连接函数

在 PHP 中使用 mysql_close()函数可以关闭使用 mysql_connect()函数打开的连接，该函数

的语法格式如下：

```
bool mysql_close ( [resource $link_identifier] )
```

mysql_close()函数的参数表示 MySQL 的连接标识符。如果没有指定，默认使用最后被 mysql_connect()打开的连接。如果没有找到该连接，函数会尝试调用 mysql_connect()建立连接并使用它。如果发生意外，系统发出 E_WARNING 级别的警告信息。下面使用该函数创建一个示例，如代码 7.16 所示。

代码 7.16　使用 mysql_close()函数关闭数据库连接

```
<?php
  $link=mysql_connect("localhost","root","0");
  if($link)
  {
     echo  "PHP 程序与数据库连接成功<br/>";
  }
  if(mysql_close($link))
  {
     echo "数据库连接已经成功关闭<br/>";
  }
  else
  {
     echo "数据库连接关闭失败";
  }
?>
```

保存并执行该段代码，结果如下：

```
PHP 程序与数据库连接成功
数据库连接已经成功关闭
```

7.3.2　单独存放连接文件

在一个大型的 Web 程序中，需要多次连接到数据库进行某种特定操作。这时再在 Web 页面中多次使用创建数据库连接的代码，就会显得很繁琐。其中的一个解决办法是将数据库连接的代码放到一个单独的文件中，其他的文件如果涉及到这个功能可以直接调用该文件。这样做的好处是方便数据库的代码修改，并达到功能相互分离和代码重用的效果。下面创建一个示例，独立 PHP 文件中的代码如下：

```
<?php
  $link=mysql_connect("localhost","root","0");
?>
```

将该文件保存，名称为 Sql-Connection.php。如果一个 PHP 页面用到连接数据库的代码，可以通过下面的代码调用：

```
<?php
  include_once("Sql-Connection.php");
?>
```

7.3.3 选择数据库

在 PHP 页面程序和 MySQL 服务器两个对象之间建立一个连接后，可以使用 mysql_select_db()函数选择数据库，然后使用 PHP 页面的脚本程序中对 MySQL 服务器中的数据库进行操作。该函数的语法格式如下：

```
bool mysql_select_db ( string $database_name [, resource $ link_identifier ] )
```

mysql_select_db()函数主要设定与指定的连接标识符所关联的服务器上的当前激活数据库。如果没有指定连接标识符，则使用上一个打开的连接。如果没有打开的连接，本函数将调用 mysql_connect()尝试打开一个新数据库连接并使用。在 PHP 中如果执行该函数成功则返回 true，否则返回 false。下面创建一个示例，具体介绍如何运用该函数选择数据库，如代码 7.17 所示。

代码 7.17 使用 mysql_select_db()函数选择数据库

```
<?php
   include_once("Sql-Connection.php");
   if(mysql_select_db("db_student"))
   {
      echo "选择数据库成功";
   }
   else
   {
      echo "选择数据库失败";
   }
   mysql_close();
?>
```

在代码 7.17 中，db_student 表示在 MySQL 服务器中创建好的数据库，保存并执行该段代码，页面执行结果如下：

```
选择数据库成功
```

7.4 数据库基本操作

在 7.3 节中讲解了如何连接数据库，以及如何选择数据库。连接并选择数据库并不能操作数据库中的数据，还需要编写并执行 SQL 语句。本节主要讲解在 PHP 中执行 SQL 语句的函

数，以及如何向数据库中插入、删除和修改数据。

7.4.1　执行 SQL 语句

PHP 程序通过使用 mysql_query()函数和 mysql_db_query()函数将 SQL 语句传递给 MySQL 数据库管理系统，然后数据库管理系统执行这些 SQL 语句，执行完毕之后返回结果。本节主要讲解这两个函数如何在 PHP 中执行数据库操作。

1. mysql_query()函数

mysql_query()函数执行一条 MySQL 查询，语法格式如下：

```
resource mysql_query ( string $query [, resource $link_identifier] )
```

mysql_query()函数表示向指定的连接标识符关联的服务器中当前活动数据库发送一条查询语句。第一个参数表示要执行的 SQL 语句，第二个参数表示资源标识符，可以用来存储该函数返回的资源。如果没有指定 link_identifier 连接标识符，则使用上一个打开的连接。如果没有打开的连接，本函数会尝试调用 mysql_connect()函数来建立一个连接并使用。

mysql_query()仅对 SELECT，SHOW，EXPLAIN 或 DESCRIBE 语句返回一个资源标识符，如果查询执行不正确则返回 false。对于其他类型的 SQL 语句，例如插入语句，mysql_query()在执行成功时返回 true，出错时返回 false。非 false 的返回值意味着查询是合法的并能够被服务器执行，但这并不说明返回的结果是影响到的行数或者返回的行数，有可能是一条查询语句执行成功了但并未影响到或并未返回任何行。

下面创建一个示例，执行一条创建数据库的语句，如代码 7.18 所示。

代码 7.18　使用 mysql_query()函数执行 SQL 语句

```
<?php
$conn = mysql_connect("localhost","root","0");
if (!$conn)
  {
  die('连接数据库失败 ' . mysql_error());
  }
$sql = "CREATE DATABASE student";
if (mysql_query($sql,$conn))
{
  echo "创建 student 数据库成功";
}
else
{
    echo "创建数据库失败<br/>" . mysql_error();
}
?>
```

保存并执行代码 7.18，页面结果如下：

```
创建 student 数据库成功
```

如果执行了一次，再执行该段代码，结果中会显示执行错误的原因。再执行代码 7.18，结果如下：

```
创建数据库失败
Can't create database 'student'; database exists
```


2. mysql_db_query()函数

mysql_db_query()函数和 mysql_query()函数的功能一样，但是该函数根据查询结果返回一个正的 MySQL 结果资源号，出错时返回 false。一般情况下，推荐使用 mysql_select_db()函数和 mysql_query()函数代替该函数的使用。mysql_db_query()函数语法格式如下：

```
resource mysql_db_query ( string $database, string $query [, resource
$ link_identifier ] )
```

mysql_db_query()函数会对 INSERT、UPDATE 和 DELETE 查询返回 true/false 来指示成功或失败。参数 database 表示进行操作的数据库对象；query 表示要执行的 SQL 语句；最后一个参数是资源标识符。该函数不会切换回先前连接到的数据库。换句话说，不能用此函数临时在另一个数据库上执行 sql 查询，只能手工切换回来。下面创建一个使用该函数的示例，如代码 7.19 所示。

代码 7.19 使用 mysql_db_query()函数

```
<?php
  $link=mysql_connect("localhost","root","0");
  if(!$link)
  {
      echo  "数据库连接失败";
  }
  $sqlText="insert into student values(1,'唐晓阳','男',23)";
  $result=mysql_db_query("db_student",$sqlText);
  echo $result;
  mysql_close();
?>
```

在代码 7.19 中，语句“$result=mysql_db_query("db_student",$sqlText);”表示执行一条插入语句，其操作的数据库对象为 db_student。

7.4.2 获取和显示数据

在 7.4.1 节中讲解了如何使用函数在数据库中执行 SQL 语句，本节将讲解如何把执行函数获得的资源标识符对象显示到页面上。在 PHP 中提供了许多这样的函数，主要有 mysql_result()函数、mysql_fetch_row()函数、mysql_fetch_array()函数、mysql_fetch_assoc()函

数和 mysql_fetch_object()函数，详细介绍如下。

1. mysql_result()函数

mysql_result()函数返回结果集中一个字段的值，在 PHP 中，如果该函数执行成功返回字段值，否则返回 false，语法格式如下：

```
mixed mysql_result ( resource $result, int $row [, mixed $field] )
```

下面创建一个示例，具体讲解如何使用该函数返回结果集中的某一个字段的值，如代码 7.20 所示。

代码 7.20　使用 mysql_result()函数

```
<?php
  $link=mysql_connect("localhost","root","0");
    if(!$link)
    {
     echo  "数据库连接失败";
    }
  mysql_select_db("db_student");
  $sqlText="select * from student";
  $result=mysql_query($sqlText);
  echo "编号: ".mysql_result($result,0,"id")."<br/>";
  echo "姓名: ".mysql_result($result,0,"name")."<br/>";
  echo "性别: ".mysql_result($result,0,"sex")."<br/>";
  echo "年龄: ".mysql_result($result,0,"age")."<br/>";
  mysql_close();
?>
```

保存并执行代码 7.20，结果如下：

```
编号: 1
姓名: 唐晓阳
性别: 男
年龄: 23
```

2. mysql_fetch_row()函数

mysql_fetch_row() 函数从结果集中取得一行数据作为数字数组，该函数的语法格式如下：

```
array mysql_fetch_row ( resource $result )
```

mysql_fetch_row()函数从 result 关联的结果集中取得一行数据并作为数组返回。每个结果的列储存在一个数组的单元中，偏移量从 0 开始。依次调用 mysql_fetch_row()将返回结果集中的下一行，如果没有更多行则返回 false。下面创建一个使用该函数的示例，如代码 7.21 所示。

代码 7.21　使用 mysql_fetch_row()函数获取结果集中的数据

```
<?php
  $link=mysql_connect("localhost","root","0");
    if(!$link)
    {
     echo  "数据库连接失败";
    }
  mysql_select_db("db_student");
  $sqlText ="select * from student";
  $result=mysql_query($sqlText);
  echo "<fieldset><legend>从数据库中获取的数据</legend>";
  echo "<table width='100%'><tr><td>编号</td><td>姓名</td><td>性别</td><td>年
  龄</td></tr>";
   while(list($id,$name,$sex,$age)=mysql_fetch_row($result))
   {
          echo "<tr><td>".$id."</td><td>".$name."</td><td>".$sex."</td>
          <td>".$age."</td></tr>";
   }
   echo "</table></fieldset>";
  mysql close();
?>
```

保存并执行代码 7.21，结果页面源代码如下：

```
<fieldset><legend>从数据库中获取的数据</legend>
<table width='100%'>
<tr><td>编号</td><td>姓名</td><td>性别</td><td>年龄</td></tr>
<tr><td>1</td><td>唐晓阳</td><td>男</td><td>23</td></tr>
<tr><td>2</td><td>裴亚敏</td><td>女</td><td>23</td></tr>
<tr><td>3</td><td>Yound Tang</td><td>男</td><td>23</td></tr>
<tr><td>4</td><td>李现蕾</td><td>女</td><td>23</td></tr>
<tr><td>5</td><td>李磊</td><td>男</td><td>23</td></tr>
<tr><td>6</td><td>江伟</td><td>男</td><td>25</td></tr>
</table></fieldset>
```

3．mysql_fetch_array()函数

mysql_fetch_array()函数从结果集中取得一行作为关联数组或数字数组或二者兼有，该函数的语法格式如下：

```
array mysql_fetch_array ( resource $result [, int $ result_type ] )
```

由该函数语法格式可知，该函数有一个可选参数 result_type，该参数有 3 个可能值。

- **MYSQL_ASSOC**　关联数组;
- **MYSQL_NUM**　数字数组;

□ **MYSQL_BOTH**　默认，同时产生关联和数字数组。

mysql_fetch_array()函数是 mysql_fetch_row()的扩展版本。除了将数据以数字索引方式储存在数组中之外，还可以将数据作为关联索引储存，用字段名作为键名。使用 mysql_fetch_array()函数获取数据库中数据，并不明显比使用 mysql_fetch_row()函数慢，并且明显提供了更多的值。下面使用该函数创建一个示例，具体代码如下。

```
<?php
    $link=mysql_connect("localhost","root","0");
    if(!$link)
    {
      echo  "数据库连接失败";
    }
  mysql_select_db("db_student");
  $exec="select * from student";
  $result=mysql_query($exec);
   print_r(mysql_fetch_array($result,MYSQL_ASSOC));
  mysql_close();
?>
```

保存并执行该段代码，结果如下：

```
Array
(
    [ID] => 1
    [name] => 唐晓阳
    [sex] => 男
    [age] => 23
)
```

4．mysql_fetch_assoc()函数

mysql_fetch_assoc()函数的返回值是根据从结果集取得的行生成的关联数组，如果连续执行该函数，并且结果集中没有下一行则返回 false。该函数语法格式如下：

```
array mysql_fetch_assoc ( resource $result )
```

下面使用该函数创建一个示例，如代码 7.22 所示。

代码 7.22　使用 mysql_fetch_assoc()函数获取结果集中的数据

```
<?php
    $link=mysql_connect("localhost","root","0");
    if(!$link)
    {
      echo  "数据库连接失败";
    }
  mysql_select_db("db_student");
```

```
    $sqlText="select * from student";
    $result = mysql_query($sqlText);
    while ($row = mysql_fetch_assoc($result))
    {
        echo "编号: ".$row['ID']."...姓名: ".$row['name']."...性别: ".$row
        ['sex0']."...年龄: ".$row['age']."<br/>";
    }
?>
```

保存并执行代码 7.22，结果如下所示：

```
编号: 1...姓名: 唐晓阳...性别: 男...年龄: 23
编号: 2...姓名: 裴亚敏...性别: 女...年龄: 23
编号: 3...姓名: Yound Tang...性别: 男...年龄: 23
编号: 4...姓名: 李现蕾...性别: 女...年龄: 23
编号: 5...姓名: 李磊...性别: 男...年龄: 23
编号: 6...姓名: 江伟...性别: 男...年龄: 25
```

5．mysql_fetch_object()

mysql_fetch_object()函数从结果集中获取一行数据作为对象，该函数如果执行成功则获得一行数据，否则返回 false。语法格式如下：

```
object mysql_fetch_object ( resource $result )
```

mysql_fetch_object()函数和 mysql_fetch_array()函数类似，只有一点区别：该函数返回的是一个对象而不是数组。间接地也意味着只能通过字段名来访问对象，而不是偏移量，并且字段名区分大小写。下面创建一个示例，具体讲解如何使用 mysql_fetch_object()函数获得数据库记录集对象，如代码 7.23 所示。

代码 7.23　使用 mysql_fetch_object()获取对象

```
<?php
  $link=mysql_connect("localhost","root","0");
    if(!$link)
    {
     echo  "数据库连接失败";
    }
  mysql_select_db("db_student");
  $sqlText="select * from student";
  $result=mysql_query($sqlText);
 while($row=mysql_fetch_object($result))
     {
       $id=$row->ID;
       $name=$row->name;
       $sex=$row->sex;
       $age=$row->age;
```

```
        echo "编号: ".$id."——姓名: ".$name."——性别: ".$sex."——年龄:
        ".$age."<br/>";
       }
    mysql_close();
?>
```


在代码 7.23 中，语句“$row=mysql_fetch_object($result)”表示获取资源对象$result 中的记录，该返回值是一个对象。如果下面没有记录，则返回 false，否则会一直向下移动。接下来利用该对象获取数值并输出。保存并执行代码 7.23，结果如下：

```
编号: 1—姓名: 唐晓阳—性别: 男—年龄: 23
编号: 2—姓名: 裴亚敏—性别: 女—年龄: 23
编号: 3—姓名: Yound Tang—性别: 男—年龄: 23
编号: 4—姓名: 李现蕾—性别: 女—年龄: 23
编号: 5—姓名: 李磊—性别: 男—年龄: 23
编号: 6—姓名: 江伟—性别: 男—年龄: 25
```

7.4.3　管理数据库数据

在一个具体的网站中，经常需要管理数据库数据，例如插入数据、删除数据、修改数据，本节主要通过示例的方式具体讲解如何在 PHP 程序中管理数据库数据。

1. 插入数据

在大型网站中，经常需要用户注册账号和填写个人相关信息。下面创建一个填写个人信息的示例，用户录入页面设计如代码 7.24 所示。

代码 7.24　用户录入页面源代码

```
<html><body style="font-family:'新宋体'; font-size:12px; margin:0;">
<center ><fieldset><table width="100%">
   <tr><td style="background-image:url(header_bg.jpg)"><img src="logo.jpg">
   </td></tr>
  </table></fieldset>
<fieldset><legend>用户录入表单</legend>
<table border=0 style=" background-color: #F9FBFD;
    border-top-width: 1px;border-top-style: solid;
    border-right-style: solid;border-bottom-style: solid;
    border-left-style: solid;border-top-color: #CDD9E7;
    border-right-color: #CDD9E7;border-bottom-color: #CDD9E7;
    border-left-color: #CDD9E7;border-right-width: 0px;
    border-bottom-width: 0px;border-left-width: 0px;   width:100%;">
<caption>学生信息输入表</caption>
<form action=insert.php method=post>
  <tr> <td align="right">编号: </td>
```

```
    <td><input type=text name=id></td>
  <tr><td align="right">姓名: </td>
    <td><input type=text name=name></td>
  <tr><td align="right">性别: </td><td><input type=text name=sex></td>
  <tr><td align="right">年龄: </td>
    <td><input type=text name=age></td>
  <tr><td></td><td>
  <input style="font-family: 'Tahoma';
   font-size: 9pt; color: #003399; border: 1px #003399 solid;color:#006699;
   BORDER-BOTTOM: #93bee2 1px solid;  BORDER-LEFT: #93bee2 1px solid;
   BORDER-RIGHT: #93bee2 1px solid;  BORDER-TOP: #93bee2 1px solid;
   background-image:url(../Images/bluebuttonbg.gif);  background-color:
   #e8f4ff;
   CURSOR: hand;  font-style: normal;" type=submit value=提交>
  </td></tr>
</form></table></fieldset></center></body></html>
```

保存代码 7.24，并重命名为 Demo-add.php，该页面实现了用户录入功能。表单提交的页面为 insert.php，下面编写该页面的 PHP 应用程序，如代码 7.25 所示。

代码 7.25　向数据库插入数据

```
<?php
  $link=mysql_connect("localhost","root","0");
   mysql_select_db("db_student");
   $id=$_POST['id'];
   $name=$_POST['name'];
   $sex=$_POST['sex'];
   $age=$_POST['age'];
   $exec="insert into student values($id,'$name','$sex',$age)";
   $result=mysql_query($exec);
   if($result)
   {
         echo "添加新学员成功";
    }
    else
    {
       echo "添加新学员失败";
    }
   mysql_close();
?>
```

保存代码 7.25，然后打开 IE 浏览器，在地址栏中输入 http://localhost/7/Demo-add.php，最后单击【转到】按钮，打开用户录入页面，效果如图 7-6 所示。

录入完表单中信息后，单击【提交】按钮，转到处理插入数据库程序页面，该页面显示用户提交信息处理结果，如下所示：

```
添加新学员成功
```



图 7-6　页面效果图

2．删除数据

在上面的示例中讲解了如何向数据库中插入数据，下面以示例的方式讲解如何删除数据库中的数据。

删除数据库数据的示例主要是从表单中获得学生编号，然后根据编号删除数据库中的数据。首先新建一个页面，然后对该页面进行布局，最后创建一个表单，并从数据库中获得数据，如代码 7.26 所示。

代码 7.26　显示数据页面代码

```
<table width="100%" cellpadding="0" cellspacing="0"
     style="background-color: #FFFFFF;
     border-top-width: 1px;  border-right-width: 1px;
     border-bottom-width: 1px;   border-left-width: 1px;
     border-top-style: solid;    border-right-style: solid;
     border-bottom-style: solid; border-left-style: solid;
     border-top-color: #0064A6;  border-right-color: #0064A6;
     border-bottom-color: #0064A6;   border-left-color: #0064A6;">
<tr>
  <td>选择</td>
  <td>编号</td>
  <td>姓名</td>
  <td>性别</td>
  <td>年龄</td>
</tr>
<?php
$link=mysql_connect("localhost","root","0");
mysql_select_db("db_student");
$exec="select * from student";
```

```
    $result=mysql_query($exec);
    while($rs=mysql_fetch_object($result))
    {
      $id=$rs->ID;
      $name=$rs->name;
      $sex=$rs->sex;
      $age=$rs->age;
      echo "<tr><td><input type=radio name=de value='$id'></td>
       <td>".$id."</td><td>".$name."</td>
       <td>".$sex."</td><td>".$age."</td></tr>";
    }
   ?>
</table>
```

在代码 7.26 中，首先获得 student 表中的数据，然后赋值给表格。保存代码 7.26，然后编写表单提交后的 PHP 处理程序，如代码 7.27 所示。

代码 7.27　删除数据库中的数据

```
<?php
  $link=mysql_connect("localhost","root","0");
    mysql_select_db("db_student");
    $id=$_POST['de'];
    $exec="delete from student where ID=$id";
    $result=mysql_query($exec);
    if((mysql_affected_rows()==0) || (mysql_affected_rows()==-1))
     {
           echo "没有找到记录，或者删除时产生错误";
           exit;
        }
    else{
    echo "该用户信息已经被删除";
    }
    mysql close();
?>
```

至此，完成了整个示例。保存代码 7.27，打开 IE 浏览器，在地址栏中输入 http://localhost/delete.php，然后单击【转到】按钮，显示数据页面如图 7-7 所示。

选择表单中要删除数据所在行的单选按钮，然后单击【删除】按钮，程序会自动转到处理删除数据程序页面，该页面显示删除数据处理结果，如下所示：

```
该用户信息已经被删除
```

3. 修改数据

在前面介绍了如何向数据库中添加数据和删除数据。下面将创建一个示例，具体讲解如

何修改数据，如代码 7.28 所示。

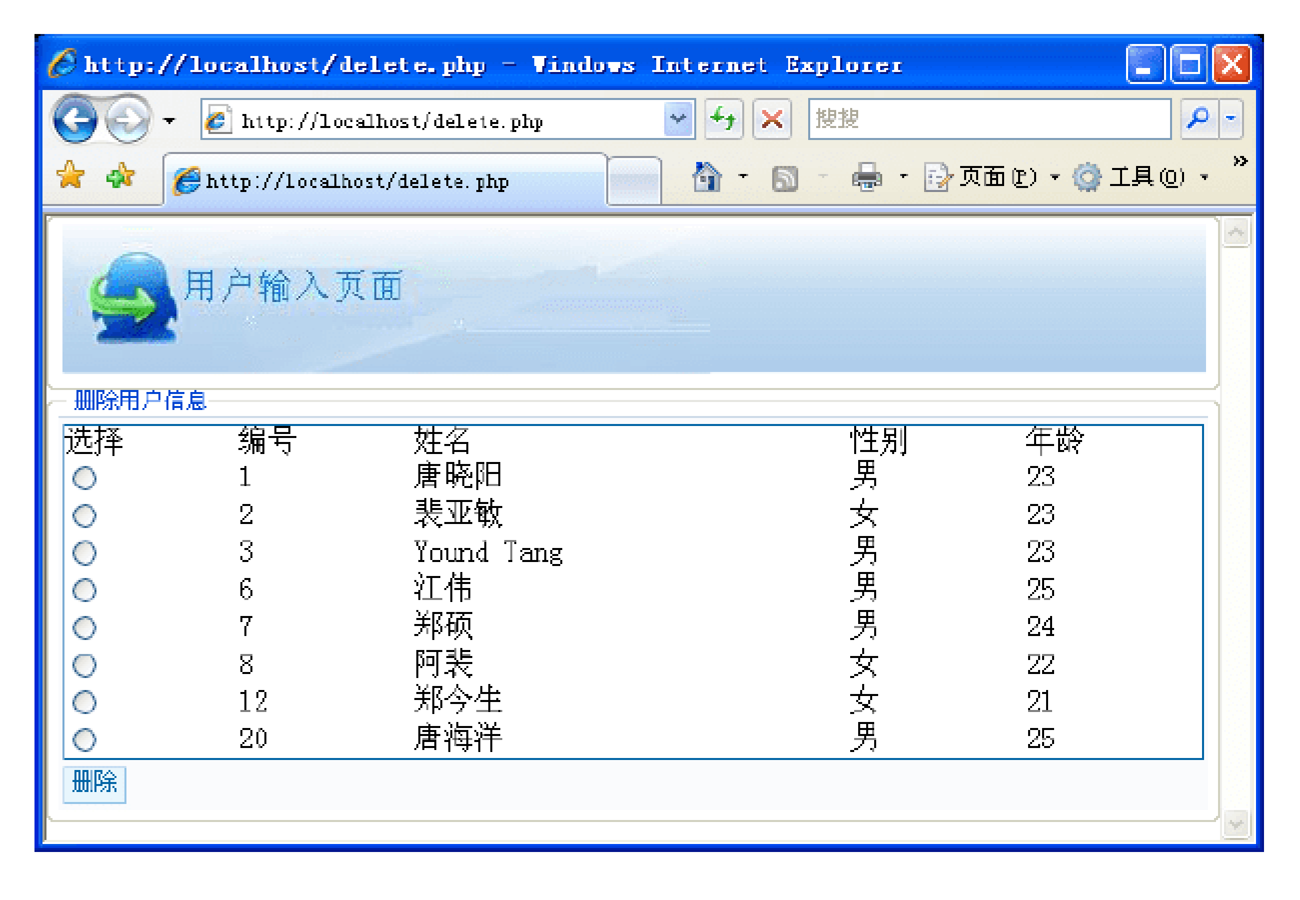


图 7-7　显示数据信息页面效果图

代码 7.28　修改数据

```
<?php
$link=mysql_connect("localhost","root","0");
    mysql_select_db("db_student");
    $id = 200812005;
    $sname ='Yound Tang';
    $sex ='men';
    $age = 27;
    $exec="update student set name ='$sname', sex='$sex',age='$age' where id=
    $id";
    if(mysql_query($exec))
    {
     echo "修改成功";
    }
    else
    {
     echo "修改失败";
    }
   ?>
```

保存并执行代码 7.28，结果如下：

```
修改成功
```

7.5　数据库高级操作

在 7.3 节和 7.4 节中，学习了如何连接数据库和对数据库数据的基本操作。只掌握了这些知识点还不能完全操作数据库，还需要学习对数据库的高级操作。例如在执行 SQL 语句出现

错误时，如何获取错误信息，还有如何获取数据库和表信息以及字段信息，同时执行多个查询或执行事务等。本节将对这些数据库高级操作进行详细介绍。

7.5.1 获取错误信息

在执行 SQL 语句时，难免会出错，出错之后应该如何解决问题呢？首先需要获得错误信息，然后再根据错误信息修改错误。在 PHP 中使用内置函数获取执行时的错误信息，一共有两个获取错误的内置函数，分别是：mysql_error()函数和 mysql_errno()函数，本节将针对这两个函数详细介绍在 PHP 中如何获取错误信息。

1. mysql_error()函数

mysql_error()函数返回上一个 MySQL 操作产生的文本错误信息，如果 MySQL 操作没有出错则返回空字符串，该函数的语法格式如下：

```
string mysql_error ( [resource $link_identifier] )
```

在使用 MySQL 进行操作时，如果没有指定连接资源号，则使用上一个成功打开的连接从 MySQL 服务器中提取错误信息。mysql_error()函数仅返回最近一次 MySQL 函数执行（不包括 mysql_error()函数和 mysql_errno()函数）的错误文本，因此如果要使用此函数，应确保在调用另一个 MySQL 函数之前检查其值。下面创建一个使用该函数的示例，具体代码如下：

```
<?php
   mysql_connect("localhost","root","0");
   mysql_select_db("MyStudent");
   echo mysql_error()."<br/>";
   $sqlText = "select * from Student";
   mysql query($sqlText);
   echo mysql error()."<br/>";
   mysql_close();
?>
```

保存并执行该段代码，结果如下：

```
Unknown database 'mystudent'
No database selected
```

2. mysql_errno()函数

mysql_errno()函数和 mysql_error()函数功能相同，只是 mysql_errno()函数返回的是错误信息的数字编码，并不返回错误的详细信息。如果执行过程没有出错，则返回 0。该函数的语法格式如下：

```
int mysql_errno ( [resource $link_identifier] )
```

下面创建一个示例，具体讲解如何运用 mysql_errno()函数。该示例使用错误的用户名和

密码连接数据库，从而返回错误信息，具体代码如下：

```
<?php
$con = mysql_connect("localhost","myroot","root");
if(!$con)
{
    echo "不能连接数据库的错误信息编号: ".mysql_errno()."<br/>";
    echo "不能连接数据库的错误信息为: ".mysql_error();
}
 ?>
```

保存并执行该段代码，页面效果如图 7-8 所示。

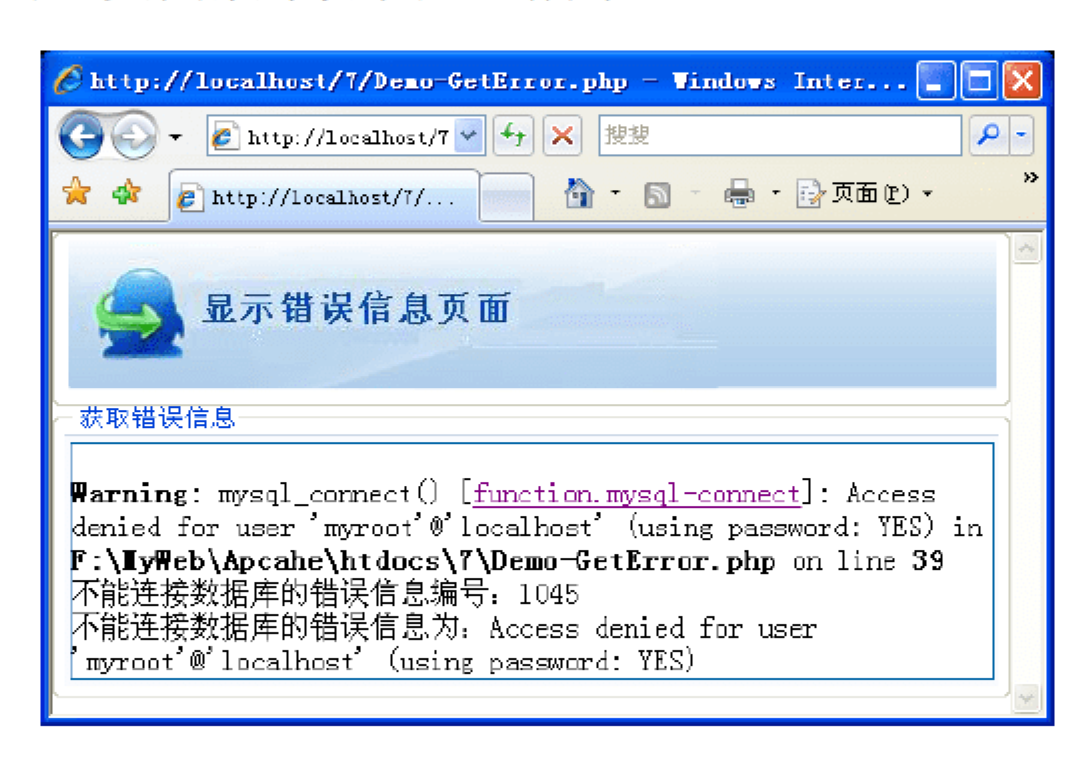


图 7-8　页面效果图

7.5.2　获取数据库和表信息

在前面的章节中，讲解了如何使用 PHP 内置函数获取数据库中的数据。同样，在 PHP 中使用内置函数也可以获取数据库和表相关信息，这些函数共有 4 个：mysql_list_dbs()函数、mysql_db_name()函数、mysql_list_tables()函数和 mysql_table_name()函数。本节将通过讲解这 4 个函数，详细介绍如何获取数据库和表信息。

1. mysql_list_dbs()函数

mysql_list_dbs()函数返回 MySQL 服务器中所有的数据库，该函数的语法格式如下：

```
resource mysql_list_dbs ( [resource $link_identifier])
```

mysql_list_dbs()函数将返回一个结果指针，包含了当前 MySQL 进程中所有可用的数据库。下面创建一个示例，具体讲解如何运用该函数，如代码 7.29 所示。

代码 7.29　使用 mysql_list_dbs()函数获取数据库信息

```
<?php
    $con = mysql_connect("localhost", "root", "0");
  if (!$con)
```

```
    {
      die('连接数据库出错原因: ' . mysql_error());
    }
    $db_list = mysql_list_dbs($con);
    while ($db = mysql_fetch_object($db_list))
    {
    echo $db->Database . "<br />";
    }
    mysql_close($con);
?>
```

保存并执行代码 7.29，效果如图 7-9 所示。

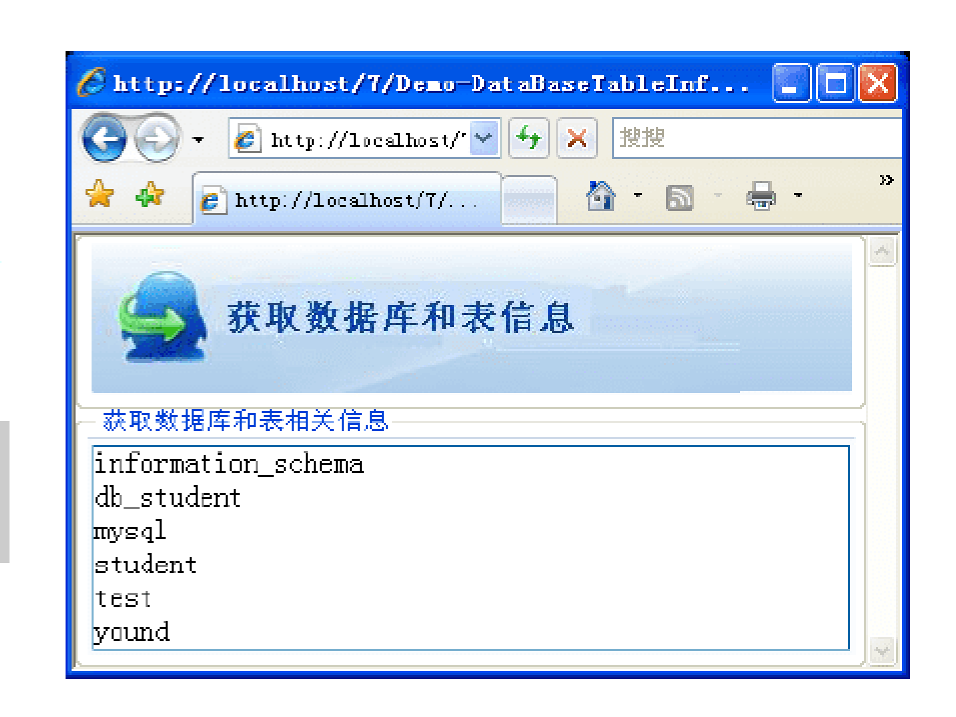


图 7-9　页面效果图

2. mysql_db_name()函数

mysql_db_name()函数取得 mysql_list_dbs()调用所返回的数据库名，该函数的语法格式如下：

```
string mysql_db_name ( resource $result, int
$row [, mixed $field] )
```

由该函数语法格式可知，参数 result 表示 mysql_list_dbs()调用所返回的结果指针。参数 row 表示结果集中的行号。参数 field 表示结果集中的字段名。如果成功则返回数据库名，失败则返回 false。如果返回 false，可以使用 mysql_error()来判断错误的种类。下面创建一个示例，具体讲解该函数的运用，如代码 7.30 所示。

代码 7.30　运用 mysql_db_name()函数

```
<?php
$con = mysql_connect("localhost", "root", "0");
if (!$con)
  {
  die('连接数据库出错原因: ' . mysql_error());
  }

$db_list = mysql_list_dbs($con);

$i = 0;
$db_count = mysql_num_rows($db_list);
while ($i < $db_count)
{
  echo mysql_db_name($db_list, $i) . "<br />";
  $i++;
}
mysql_close($con);
?>
```

保存并执行代码 7.30，页面效果如图 7-9 所示。

3．mysql_list_tables()函数和 mysql_table_name()函数

mysql_list_tables()接受一个数据库名并返回和 mysql_query()函数很相似的一个结果指针。用 mysql_table_name()函数来遍历此结果指针，或者任何使用结果表的函数。mysql_table_name()接受 mysql_list_tables()返回的结果指针，返回表名。这两个函数一般不推荐使用，所以在此就不再详细讲解。

7.5.3　获取字段信息

在 PHP 中，有一组内置函数专门用于获取 MySQL 服务器数据库中表字段的相关信息，例如字段名称、字段数据类型和是否为主键等。本节主要介绍这些获取字段信息的内置函数。

1．mysql_fetch_field()函数

mysql_fetch_field()函数从结果集中取得列信息并作为对象返回，该函数的语法格式如下：

```
object mysql_fetch_field ( resource $result [, int $field_offset] )
```

mysql_fetch_field()函数可以用来从某个查询结果中取得字段的信息。参数 field_offset 表示字段偏移量，如果没有指定字段偏移量则下一个尚未被 mysql_fetch_field()取得的字段被提取。该函数返回的对象属性为字段的相关信息，例如列名和该列所在表的表名，这些属性如下。

- **name**　列名;
- **table**　该列所在的表名;
- **def**　该列的默认值;
- **max_length**　该列最大长度;
- **not_null**　如果该列不能为 NULL，返回 1，否则返回 0;
- **primary_key**　如果该列是主键，返回 1，否则返回 0;
- **unique_key**　如果该列是 unique key，返回 1，否则返回 0;
- **multiple_key**　如果该列是 non-unique key，返回 1，否则返回 0;
- **numeric**　如果该列是 numeric，返回 1，否则返回 0;
- **blob**　如果该列是 BLOB，返回 1，否则返回 0;
- **type**　该列的类型;
- **unsigned**　如果该列是无符号数，返回 1，否则返回 0;
- **zerofill**　如果该列是 zero-filled，返回 1，否则返回 0。

下面创建一个示例，具体讲解如何运用该函数，如代码 7.31 所示。

代码 7.31　使用 mysql_fetch_field()函数获取列相关信息

```
<?php
$con = mysql_connect("localhost", "root", "0");
```

```
    if (!$con)
    {
      die('连接数据库出错的原因： ' . mysql_error());
    }
    $db_selected = mysql_select_db("db_student",$con);
    $sql = "SELECT * from student";
    $result = mysql_query($sql,$con);
    echo "<table><tr>";
    while ($property = mysql_fetch_field($result))
    {
      echo "<td>";
      echo "列名： " . $property->name . "<br />";
      echo "表名： " . $property->table . "<br />";
      echo "Default value: " . $property->def . "<br />";
      echo "Max length: " . $property->max_length . "<br />";
      echo "Not NULL: " . $property->not_null . "<br />";
      echo "Primary Key: " . $property->primary_key . "<br />";
      echo "Unique Key: " . $property->unique_key . "<br />";
      echo "Multiple Key: " . $property->multiple_key . "<br />";
      echo "Numeric Field: " . $property->numeric . "<br />";
      echo "BLOB: " . $property->blob . "<br />";
      echo "Field Type: " . $property->type . "<br />";
      echo "Unsigned: " . $property->unsigned . "<br />";
      echo "Zero-filled: " . $property->zerofill . "<br /><br />";
      echo "</td>";
    }
    echo "</tr></table>";
    mysql_close($con);
    ?>
```

在代码 7.31 中，运用 mysql_fetch_field()函数返回的对象，获取了字段的相关属性。保存并执行代码 7.31，页面效果如图 7-10 所示。

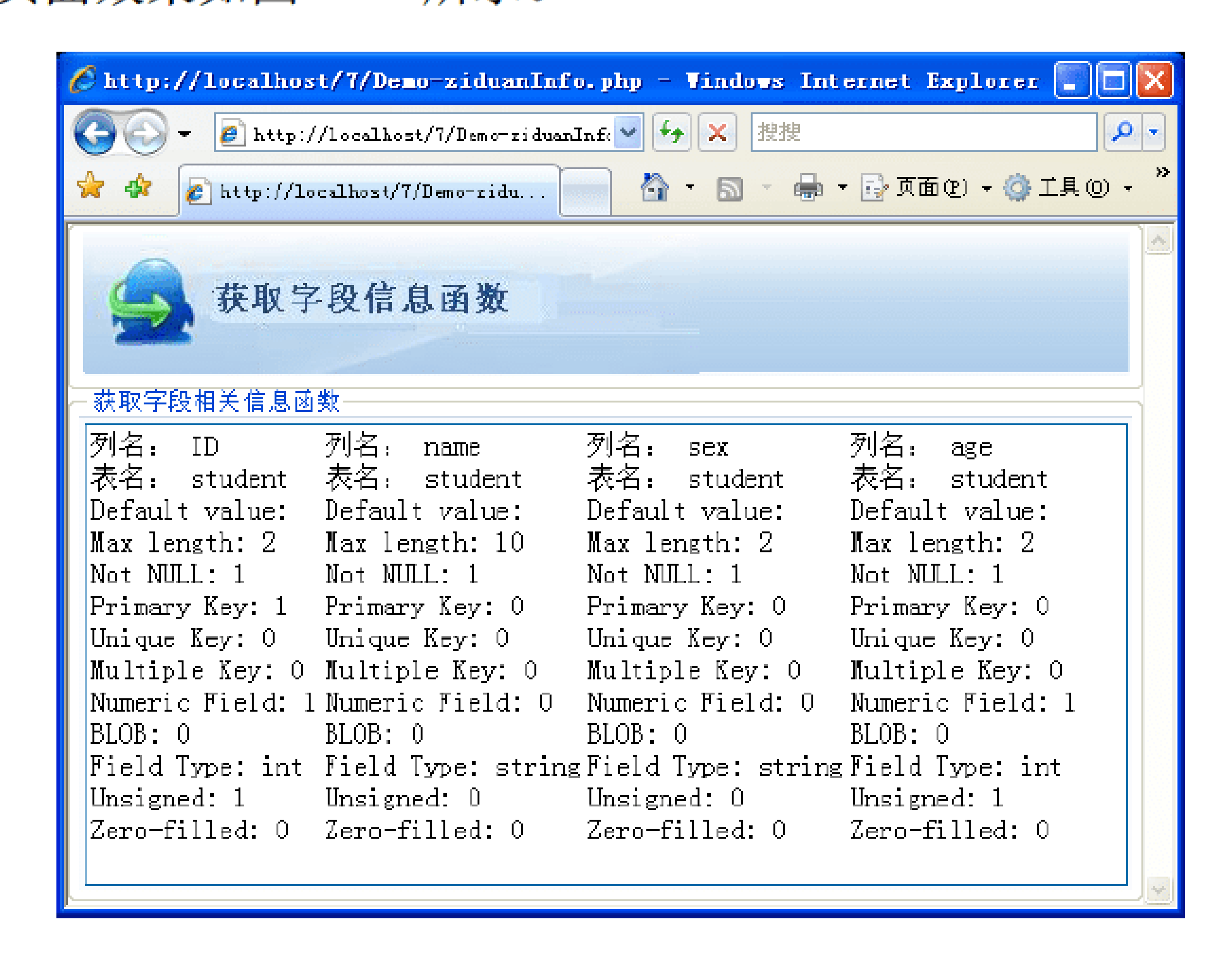


图 7-10　运用 mysql_fetch_field()函数示例页面效果图

2．mysql_num_fields()函数

mysql_num_fields() 函数返回结果集中字段的数量，该函数的语法格式如下：

```
int mysql_num_fields ( resource $result )
```

下面创建一个示例，具体讲解如何使用该函数获得字段的数量，代码如下：

```
<?php
$con = mysql_connect("localhost", "root", "0");
if (!$con)
{
  die('连接数据库出错的原因： ' . mysql_error());
}
mysql_select_db("db_student");
$result = mysql_query("select * from student");
echo "表 student 中共有".mysql_num_fields($result)."个字段";
mysql_close();
?>
```

保存并执行该段代码，结果如下：

```
表 student 中共有 4 个字段
```

3．mysql_list_fields()函数

mysql_list_fields()函数表示获取给定表名的信息。语法格式如下：

```
resource mysql_list_fields ( string $database_name, string $table_name [,
resource $link_identifier] )
```

该函数的参数是数据库名和表名，返回一个结果指针，可以用于 mysql_field_flags()函数、mysql_field_len()函数、mysql_field_name()函数和 mysql_field_type()函数等。mysql_list_fields()函数的使用示例如下：

```
<?php
$con = mysql_connect('localhost', 'root', '0');
$fields = mysql_list_fields("db_student", "student", $con);
$columns = mysql_num_fields($fields);
for ($i = 0; $i < $columns; $i++)
{
    echo mysql_field_name($fields, $i) . "<br/>";
}
?>
```

4．mysql_field_flags()函数

mysql_field_flags()函数从结果中取得和指定字段关联的标志，例如 primary-key、enum 和

binary 等。该函数语法格式如下：

```
string mysql_field_flags ( resource $result, int $field_offset )
```

参数 field_offset 表示从哪个字段开始，0 指示第一个字段。下面创建一个示例，具体讲解如何运用 mysql_field_flags()函数，代码如下：

```
<?php
$con = mysql_connect("localhost", "root", "0");
if (!$con)
{
  die('连接数据库失败的原因是:  ' . mysql_error());
}
$db_selected = mysql_select_db("db_student",$con);
$sql = "SELECT * from student";
$result = mysql_query($sql,$con);
$flags = mysql_field_flags($result, 0);
echo $flags;
mysql_close($con);
?>
```

保存并执行该段代码，结果如下：

```
not_null primary_key unsigned auto_increment
```

5. mysql_field_len()函数

该函数返回指定字段的长度，如果失败则返回 false，语法格式如下：

```
int mysql_field_len ( resource $result, int $field_offset )
```

由于该函数以及以下要讲解的函数都比较简单，所以在讲完所有的函数后创建一个综合示例，具体讲解这些函数。

6. mysql_field_name()函数

mysql_field_name()返回指定索引的字段名，语法格式如下：

```
string mysql_field_name ( resource $result, int $field_index )
```

result 必须是一个合法的结果标识符，field_index 是该字段数字偏移量，从 0 开始。

7. mysql_field_type()函数

mysql_field_type()函数和 mysql_field_name()函数相似。参数完全相同，但 mysql_field_type()函数返回的是字段类型而不是字段名。字段类型有“int”、“real”、“string”和“blob”以及其他。该函数的语法格式如下：

```
string mysql_field_type ( resource $result, int $field_offset )
```

8．mysql_field_table()函数

mysql_field_table()函数返回指定 result 结果集中指定偏移量的字段所在表的表名。该函语法格式如下：

```
string mysql_field_table ( resource $result, int $field_offset )
```

通过前面的学习，读者应该可以运用这些函数获取字段的相关信息，下面创建一个综合示例，具体说明这些函数的运用，如代码 7.32 所示。

代码 7.32　综合示例

```
<?php
$con = mysql_connect("localhost", "root", "0");
if (!$con)
{
  die('连接数据库失败的原因是： ' . mysql_error());
}
mysql_select_db("db_student",$con);
$sql = "select * from student";
$result = mysql_query($sql);
$zd = mysql_field_name($result,1);
echo "字段".$zd."的长度是： ".mysql_field_len($result,1)."<br/>";
echo "字段".$zd."的类型是： ".mysql_field_type($result,1)."<br/>";
echo "字段".$zd."所在的表名： ".$tableName=mysql_field_table($result,1);
echo "<br/>";
echo "表".$tableName."共有<font color='red'>".mysql_num_fields($result).
"</font>个字段<br/>";
$columns= mysql_list_fields("db_student","student",$con);
for ($i = 0; $i < $columns; $i++)
{
    echo "字段".mysql_field_name($result, $i) . "关联的标志是： ".mysql_field_
    flags($result,$i)."<br/>";
}
mysql_close($con);
?>
```

在代码 7.32 中，使用了本节所讲解函数中除第一个函数外的所有函数，通过该示例进一步地学习和更加灵活地运用这些函数，保存并执行代码 7.32，页面运行效果如图 7-11 所示。

7.5.4　辅助函数

在 PHP 中可以很轻松地获取客户端或 MySQL 服务器端信息，这是因为 PHP 提供了许多相关的内置函数，本节将详细介绍这些函数。

1．mysql_client_encoding()函数

mysql_client_encoding()函数从 MySQL 中获取字符集的名称，语法格式如下：

```
string mysql_client_encoding ([resource $link_identifier] )
```

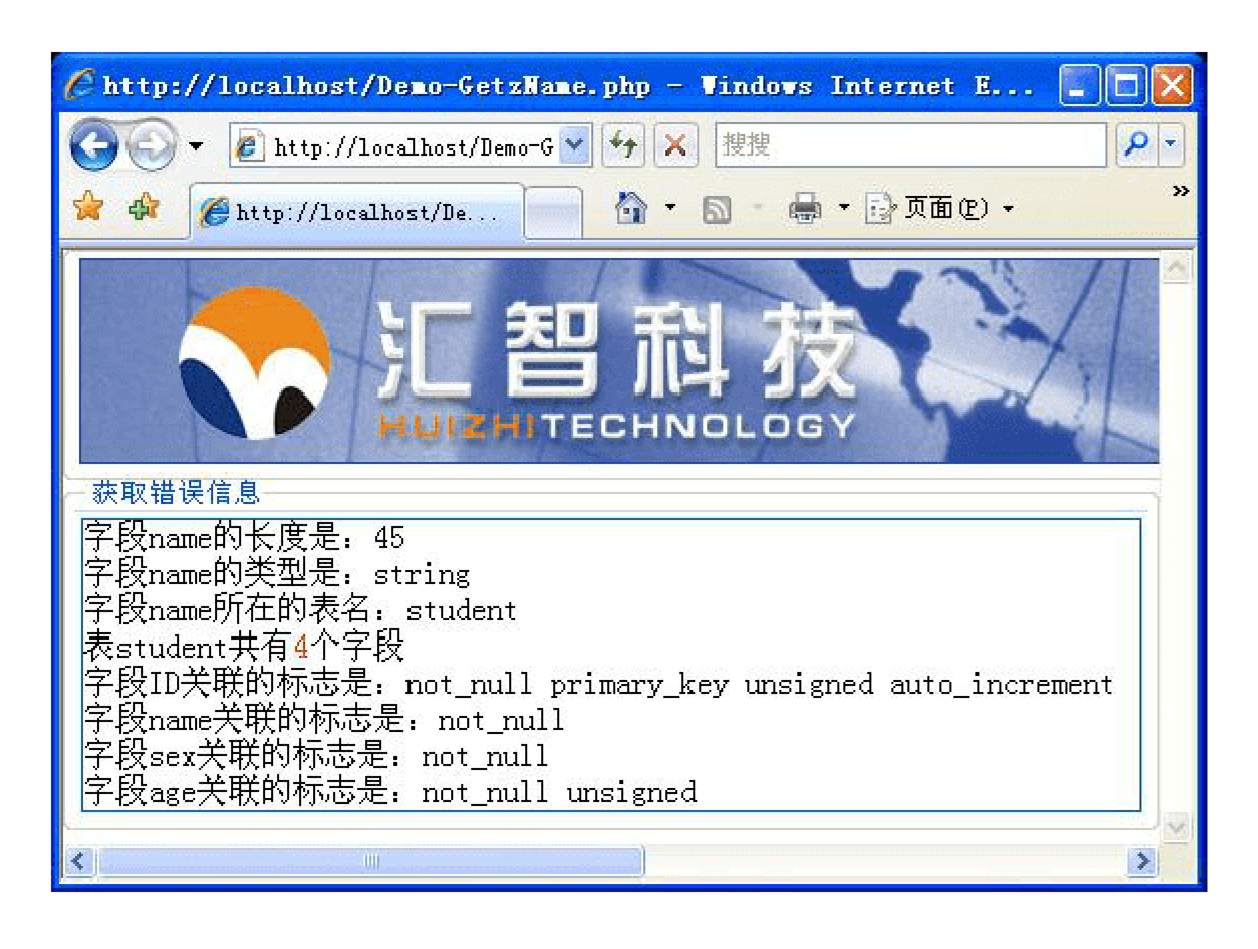


图 7-11　页面效果图

2．mysql_get_server_info()函数

mysql_get_server_info()函数返回 link_identifier 所使用的服务器版本信息，语法格式如下：

```
string mysql_get_server_info ( [resource $link_identifier] )
```

3．mysql_get_host_info()函数

mysql_get_host_info()函数返回与 MySQL 连接所使用的连接方式，包括服务器的主机名，该函数的语法格式如下：

```
string mysql_get_host_info ( [resource $link_identifier] )
```

4．mysql_get_client_info()函数

mysql_get_client_info()函数返回客户端库的版本，语法格式如下：

```
string mysql_get_client_info ( void )
```

5．mysql_stat()函数

mysql_stat()函数返回当前服务器状态，例如正常运行时间、线程、执行的查询、太慢的查询、已打开的表、已刷新的表、当前打开的表和每秒平均查询数等。如果要获得其他状态变量的完整列表，只能使用 SQL 命令 SHOW STATUS。该函数的语法格式如下：

```
string mysql_stat ( [resource $link_identifier] )
```

通过前面的学习，读者对这些函数也有了一定的了解，下面创建一个示例，具体讲解在 PHP 程序中如何运用这些函数，如代码 7.33 所示。

代码 7.33　获取客户端或 MySQL 服务器端信息

```
<?php
$con = mysql_connect("localhost", "root", "0");
if (!$con)
{
  die('连接数据库失败的原因是： ' . mysql_error());
}
echo "当前使用的字符编码集为: ".mysql_client_encoding($con)."<br/>";
printf("MySQL 服务器版本为: ".mysql_get_server_info($con)."<br/>");
printf ("连接方式: %s<br/>", mysql_get_host_info());
printf ("客户端版本: %s\n", mysql_get_client_info());
echo "<br/>";
printf("服务器状态: ".mysql_stat());
?>
```

保存并执行代码 7.33，页面效果如图 7-12 所示。

7.5.5　多个查询

在 PHP 中可以执行多个查询，该功能是在新的 MySQLi 扩展中新提供的一个特性。PHP 可以执行多个查询，并且在需要时获取每个结果集，本节介绍其实现方法。



图 7-12　页面效果图

在 PHP 中使用 mysqli_multi_query()函数可以连续执行一个或多个查询，使程序员可以一次执行多个查询，然后在必要时，获取每个结果集。该函数还有两个辅助函数：mysqli_more_results() 函数和 mysqli_next_result()函数。mysqli_more_results()函数确定 mysqli_multi_query()调用返回的结果集中是否还有其他结果；mysqli_next_result()函数从 mysqli_multi_query()函数返回的结果集中获取下一个结果。下面创建一个示例，详细介绍这 3 个函数的使用方法和如何执行多个查询语句，如代码 7.34 所示。

代码 7.34　多个查询

```
<?php
 $con = mysqli_connect("localhost","root","0");
 if(!$con)
 {
    echo "连接 MySQL 数据库出错原因: ".mysqli_connect_error();
 }
 $con->select_db("db_student");
 $query = "select ID, name from student;";
```

```
 $query .= "select sex, count(*) from student group by sex ";
 echo "<table><tr>";
 $i=1;
if(mysqli_multi_query($con,$query))
{
    do
    {
        $result =mysqli store result($con);
        echo "<td valign='top'><li>现在执行的是多个查询语句中的第".$i."个查询语句:
        </li><br/>";
        while ($row = mysqli_fetch_row($result))
        {
            echo $row[0]."——".$row[1]."<br/>";
        }
         if(mysqli_more_results($con))
        {
        $i++;
        echo "</td>";
        }

    }while (mysqli next result($con));
}
echo "</tr></table>";
mysqli_close($con);
?>
```

在代码 7.34 中，运用的是 MySQLi 扩展程序，因此在连接数据库时也使用扩展程序中连接数据库的函数。保存并执行代码 7.34，页面效果如图 7-13 所示。



图 7-13　页面效果图

7.5.6　准备语句

通常会重复执行一条 SQL 命令，每次执行 SQL 语句都要进行编译和用户验证，这样实现

起来开销非常大，并且编写代码也不方便。为帮助解决重复执行查询带来的问题，MySQL 引入了准备语句，可以以低开销和更少的代码实现上述任务。有两种准备语句。

- **绑定参数**　绑定参数准备语句允许将查询存储在 MySQL 服务器上，只将迭代数据重复地发送给服务器，再将这些迭代数据集成到查询中执行。例如，有一个需要录入学生档案的 Web 应用程序，可以创建一个 Web 表单，接受最多 10 个学生的编号、姓名和性别，因为此信息使用相同的 SQL 语句插入数据库中，所以可以使用绑定参数准备语句。
- **绑定结果**　绑定结果准备语句允许将 PHP 变量绑定到所获取的相应字段，从而使有时难以处理的索引数组或关联数组从结果中提取值，然后在必要时使用这些变量。

下面将详细介绍使用绑定参数和绑定结果语句所使用到的方法，以及如何运用绑定参数和绑定结果准备语句。

1．绑定参数

使用绑定参数准备语句执行 SQL 语句时，用到的方法有 mysqli_stmt_prepare()方法、mysqli_stmt_execute()方法、mysqli_stmt_close()方法和 mysqli_stmt_bind_param()方法，这些方法的介绍如表 7-5 所示。

表 7-5　准备语句方法

方法名称	过程化语法格式	面向对象语法格式	参数描述	功能说明

mysqli_stmt_prepare()	boolean
	mysqli_stmt_prepare
	(
	mysql_stmt stmt
	)
		class mysqli_stmt
		{
		Boolean prepare()
		}
			stmt 表示一个准备语句对象	准备要执行的语句

mysqli_stmt_execute()	Boolean
	mysqli_stmt_execute
	(
	mysqli_stmt stmt
	)
		class stmt
		{
		Boolean execute()
		}
			stmt 表示一个准备语句对象	执行准备语句，该语句何时执行，取决于语句类型

mysqli_stmt_close()	boolean
	mysqli_stmt_close
	(
	mysql_stmt stmt
	)
		class stmt
		{
		Boolean close()
		}
			同上	关闭准备语句占用的资源

mysqli_stmt_bind_param()	boolean
	mysqli_stmt_bind_param
	(
	mysqli_stmt stmt,string types,mixed &varl [,mixed &varN]
	)
		class stmt
		{
		Boolean bind_param()
		}
			stmt 表示一个准备语句对象；
			types 参数表示其后各个变量（$var1...$varn）的数据类型
				将变量名绑定到相应的字段（绑定参数）

由表 7-5 知，mysqli_stmt_prepare()方法表示准备要执行的语句，有一点要说明，无论使用绑定参数还是使用绑定结果的准备语句，都需要使用该方法准备要执行的语句。下面创建一个示例，具体说明绑定参数如何执行 SQL 语句，如代码 7.35 所示。

代码 7.35　绑定参数准备语句

```
<?php
$mysqli = new mysqli("localhost","root","0","db_student");
$query = "insert into student values (ID=?,name=?,sex=?,age=?)";
$stmt = $mysqli->stmt_init();
$stmt->prepare($query);
$stmt->bind_param('issi',$id,$name,$sex,$age);
$idArray=$_POST['id'];
$nameArray=$_POST['name'];
$sexArray=$_POST['sex'];
$ageArray=$_POST['age'];
$x=0;
while ($x<sizeof($idArray)) {
    $id =$idArray[$x];
    $name =$nameArray[$x];
    $sex = $sexArray[$x];
    $age=$ageArray[$x];
    $stmt->execute();
}
$stmt->close();
$mysqli->close();
?>
```

在代码 7.35 中，“？”代表一个占位符，即后面插入学生档案的编号、姓名、性别和年龄。在下面调用$stmt->bind_param()方法，以方法中出现的相同顺序将变量$id、$name、$sex、$age 绑定到问号表示的字段占位符。此时每个数据行都已准备好，使用$stmt->execute()方法发送给服务器处理。最后，关闭并回收占用资源。

2. 绑定结果准备语句

使用绑定结果准备语句执行 SQL 语句时，用到的方法有 mysqli_stmt_prepare()方法、mysqli_stmt_bind_result()方法和 mysqli_stmt_fetch()方法，由于第一个方法在表 7-5 中已经介绍，在此不再赘述，下面介绍 mysqli_stmt_bind_result()方法和 mysqli_stmt_fetch()方法。

mysqli_stmt_bind_result()方法将变量绑定到所获取的字段上，该函数的过程化语法格式如下：

```
boolean mysqli_stmt_bind_result(mysqli_stmt stmt,mixed &var1 [,mixed
&varN...])
```

该函数的面向对象语法如下：

```
class mysqli_stmt
{
boolean bind_result(mixed &var1[,mixed &varn])
}
```

下面创建一个示例，使用绑定结果准备语句查询数据，如代码 7.36 所示。

代码 7.36　使用绑定结果查询语句

```
<?php
  $link=mysqli_connect("localhost","root","0");
  if(!$link)
      echo mysqli_connect_error()."所在位置为".mysqli_connect_errno();
  $link->select_db("db_student") or die("找不到该数据库");
  $query="select name,sex from student order by sex ";
  $stmt=$link->stmt_init();
  $stmt->prepare($query);
  $stmt->execute();
  $stmt->bind_result($name,$sex);
  while($stmt->fetch()){
       echo "$name,$sex";
   }
  $stmt->close();
  $link->close();
?>
```

在代码 7.36 中，查询语句中字段个数要和$stmt->bind_result($name,$sex)绑定参数的个数相同。$stmt->fetch()方法主要获取准备语句结果的每条记录，并将相应字段赋给绑定结果。

7.5.7　事务处理

事务是作为一个单元的、一组有序的数据库操作。如果组中的所有操作都成功，则认为事务成功，即使只有一个操作失败，事务也不成功。如果所有的操作都成功则事务提交（commit），将作用于其他数据库进程，如果一个操作失败，则事务回滚（roll back），该事务所有影响将取消。

事务是数据库的实现，而非语言的实现，语言的实现只不过是包装了数据库提供的 API 或者直接使用数据库所提供的事务相关的 SQL Statement。在 PHP 5.0 中可以调用 MySQL 的 API 来实现事务的处理。PHP 主要通过下列函数实现事务处理，分别为 autocommit()、commit()和 rollback()。下面详细介绍这些方法。

1. autocommit()函数

autocommit()函数控制 MySQL 自动提交模式的行为，该函数的过程化语法格式如下所示：

```
boolean autocommit(boolean  mode)
```

autocommit()函数面向对象语法格式如下：

```
class mysqli
{
bool autocommit ( bool $mode )
}
```

2. commit()函数

commit()函数将当前事务提交给数据库，成功时返回 true，否则返回 false。该函数的过程化语法格式如下：

```
boolean commit()
```

commit()函数面向对象语法格式如下：

```
class mysqli
{
bool commit ( void )
}
```

3. rollback()函数

rollback()函数回滚当前事务，成功时返回 true，否则返回 false。该函数的过程化语法格式如下：

```
boolean rollback()
```

rollback()函数面向对象语法格式如下：

```
class mysqli
{
bool rollback ( void )
}
```

下面使用这些函数创建一个使用事务的示例，如代码 7.37 所示。

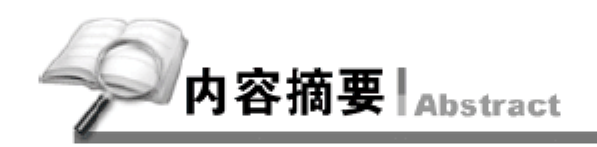
代码 7.37　事务示例

```
<?php
 $con = new mysqli("localhost","root","0");
 if(!$con)
 {
    echo mysqli_error();
 }
 $con->select_db("db_student");
 $success= true;
 $con->autocommit(false);
 $query ="insert into student values(20,'杨阳','男',25)";
```

```
    $result = $con->query($query);
    if(!$result)
    {
       $success =false;
    }
    $query ="update student set name= '唐海洋' where id =20";
    $result = $con ->query($query);
    if(!$result)
    {
       $success = false;
    }
    if($success)
    {
       $con->commit();
       echo "事务提交成功";
    }
    else
    {
       $con->rollback();
       echo "事务提交失败";
    }
    $con->autocommit(true);
?>
```

在代码 7.37 中，在事务的每个步骤执行之后都会检查查询状态和受影响的记录。如果在任何时刻失败，$success 都将设置为 false，所有的步骤都会在脚本结束时回滚。当然还可以优化此脚本，以锁步的方式启动各个查询，即每个查询只在确定了前面的查询确实正确执行后才发生。

第8章 在 PHP 中处理 XML



XML 是一种标准化的数据格式，它类似于 HTML，都有标签和实体。但是 XML 又不完全和 HTML 一样，XML 可用于远程调用，在程序中可以通过解压缩信息或者用 HTML 显示的方式使用 XML 文件。大部分语言都支持 XML，例如 Java、C#和 C++等。PHP 对 XML 的支持更加强大，不仅实现了 XML 所有流行接口，例如 DOM 和 SAX 等，还内置 SimpleXML 函数库，方便初学者解析 XML 文档。本章主要介绍如何使用捆绑在 PHP 中的 XML 解析器，以及在客户端处理 XML。

学习目标 | Objective

- 掌握 PHP 如何生成 XML 文档
- 了解 PHP 解析 XML 文档的几种方法
- 掌握 SAX 解析器如何解析 XML 文档
- 掌握使用 DOM 库操作 XML 文件
- 掌握 SimpleXML 处理 XML 文件
- 掌握 SimpleXML 所支持的函数
- 了解 JavaScript 如何处理 XML 文件

8.1 PHP 生成 XML

通过前面章节的学习可知，在 PHP 中可以动态地生成 HTML，同样也可以生成动态的 XML 文档。动态 XML 的一个典型应用是 RSS，一种用来同步新闻站点的文件格式。PHP 可以读取数据库或者 HTML 文件中的文章信息，生成一个基于这些信息的 XML 摘要文件，本节主要讲解如何运用 PHP 生成动态的 XML。下面创建一个示例，如代码 8.1 所示。

代码 8.1　生成动态的 XML 文档

```
<?php header('Content-Type: text/xml');
echo "<?xml version='1.0' encoding='gb2312' ?>";
?>
<rss version="0.91">
```

```
    <channel>
      <?php
       $items = array(
                    array('title' => 'PHP 程序设计',
                          'link'  => 'http://www.itzcn.com',
                          'bookAbstract'  =>'本书主要从 PHP 基础讲解，特别对初学者有很
                          大帮助，对于比较熟练的程序员也有技术上的帮助'),
                    array('title' => 'AJAX 从入门到精通',
                          'link'  => 'http://www.itzcn.com',
                          'bookAbstract'  => '本书主要讲解 AJAX 基础技术，以及 AJAX 的
                          高级应用')
                    );

       foreach($items as $item) {
         echo "<item>\n";
         echo "  <title>{$item[title]}</title>\n";
         echo "  <link>{$item[link]}</link>\n";
         echo "  <bookAbstract>{$item[bookAbstract]}</bookAbstract>\n";
         echo "  <language>en-us</language>\n";
         echo "</item>\n";
       }
      ?>
    </channel>
  </rss>
```

由代码 8.1 知，在 PHP 中生成动态 XML 文档很简单。只需要用 header()函数将文档的 MIME 类型改成"text/ xml"，然后用 PHP 生成一个 RSS 文档。RSS 文档是一个包含有若干个 channel 元素的 XML 文档，每个 channel 包含有几个 item 元素，每个 item 元素又包含有一个 title、一个 bookAbstract 和一个 link 元素。PHP 生成 XML 就像生成 HTML 一样，只需要用 echo 打印内容即可，为了避免<?xml ... ?>声明被解释为一个 PHP 标签，需要编辑 php.ini 配置文件，将 short_open_tag 选项设为不启用，或者直接用 echo 将这一行打印出来。

```
<?php echo '<?xml version="1.0" encoding="ISO-8859-1" ?>'; ?>
```

执行代码 8.1，PHP 生成的 XML 文档如代码 8.2 所示。

代码 8.2　PHP 生成的动态 XML 文档

```
  <?xml version="1.0" encoding="gb2312" ?>
- <rss version="0.91">
- <channel>
-   <item>
      <title>PHP 程序设计</title>
      <link>http://www.itzcn.com</link>
      <bookAbstract>本书主要从 PHP 基础讲解，特别对初学者有很大帮助，对于比较熟练的程序
      员也有技术上的帮助</bookAbstract>
```

```
        <language>en-us</language>
    </item>
-     <item>
        <title>AJAX 从入门到精通</title>
        <link>http://www.itzcn.com</link>
        <bookAbstract>本书主要讲解 AJAX 基础技术，以及 AJAX 的高级应用</bookAbstract>
        <language>en-us</language>
      </item>
    </channel>
    </rss>
```

8.2 PHP 处理 XML

使用 PHP 技术可以操作 XML 文档。客户端将函数名和参数值用 XML 编码后，通过 HTTP 发送至服务器，服务器收到后解码函数名和值，再决定如何处理，最后返回一个 XML 编码的响应值。在 PHP 中处理 XML 最常见的操作有：获取数据、添加数据、修改数据和删除数据等。这些操作都是通过实现 XML 接口 SAX 和 DOM 中的方法而进行的。本节将对这两个接口进行分析，并使用 DOM 接口对 XML 文档进行相应的操作。

8.2.1 解析 XML 文档方法比较

在 PHP 中，可以使用正则表达式和字符串函数解析 XML 文档，可以使用 DOM 库对 XML 文档进行解析，可以使用基于 Expat C 库的事件驱动型解析器（即 SAX 解析器）解析 XML 文档，还可以使用适合于解析简单 XML 文件的解析器 SimpleXML，本节主要对这些解析 XML 文档的方法进行介绍和比较。

1. 使用正则表达式和字符串函数解析 XML 文档

使用正则表达式代码读取 XML 文档，必须确保 XML 的格式良好，这样才可以顺利地读取，但是有些格式正确的 XML 可能与正则表达式不匹配，所以一般不建议使用正则表达式读取 XML。但是它是兼容性最好的方式，因为正则表达式函数总是可用。不要用正则表达式直接读取来自用户的 XML，因为无法控制这类 XML 的格式或结构。

2. 使用 DOM 库对 XML 文档进行解析

DOM 是 Document Object Model 的缩写，使用 DOM 接口处理 XML 文档，其机制如下。需要将整个 XML 文件加载到内存中去，并以一棵节点树的形式存在。当完成加载后，可以对该节点树中的每一个节点进行操作。换句话说，通过 DOM 树，应用程序可以对 XML 文档进行随机访问。这种访问方式给应用程序的开发带来了很大的灵活性，它可以任意地控制整个 XML 文档中的内容，从而执行添加、删除和修改等操作。

由于 DOM 解析器将整个 XML 文档转化成 DOM 树放在内存中，因此当 XML 文档比较

大或者文档结构比较复杂时，对内存的需求就比较高。而且对于结构复杂的树的遍历也是一项比较耗时的操作。所以 DOM 解析器对机器性能的要求比较高，实现效率不十分理想。不过 DOM 解析器的树结构的思想与 XML 文档的结构相吻合，而且通过 DOM 树机制很容易实现随机访问。

3．使用 SAX 解析器解析 XML 文档

读取 XML 的另一种方法是使用 XML Simple API（SAX）解析器，这也是最常用的解析器。在解析时不要求验证 XML 文档，就可以找出当前的 XML 标签及其包含的内容，但是却不能确定它们是否为正确文档结构中的正确标签。不过在实际中，通常这不是大问题。SAX 解析器运行在回调模型上，每次打开或关闭一个标记时，或者每次解析器看到文本时，就用节点或文本的信息回调用户定义的函数。SAX 解析器的优点是它是真正轻量级的解析器，不会在内存中长期保持内容，所以可以用于非常巨大的文件。缺点是编写 SAX 解析器回调非常麻烦。

同 DOM 分析器相比，SAX 解析器对 XML 文档的处理缺乏一定的灵活性，然而对于那些只需要访问 XML 文档中的数据而不对文档进行更改的应用程序来说，SAX 解析器的效率则更高。由于 SAX 解析器实现简单，对内存要求比较低，因此实现效率比较高。

4．使用解析器 SimpleXML 解析 XML 文档

如果解析非常简单的 XML 文档，推荐使用 SimpleXML 解析器。SimpleXML 解析 XML 文档非常容易，但是 SimpleXML 无法像 DOM 扩展那样生成 XML 文档，也不像 Expat 扩展那样灵活可扩展且高效。

8.2.2　SAX 解析器解析 XML

SAX 解析器是基于 Expat C 库的事件驱动型解析器，该 PHP 解析器也是最常用的解析器之一。由于 SAX 解析器基于事件，所以在解析时不要求验证 XML 文档，可以直接找出当前的 XML 标签及其包含的内容，但是却不能确定它们是否为正确文档结构中的正确标签。不过在实际中，该问题可以不考虑。

当 SAX 解析器阅读文档时，将针对各种事件调用不同的处理程序。本节将主要围绕 SAX 解析器讨论可以提供的处理程序，以及设置这些处理程序的函数和触发调用处理程序的事件。

1．元素处理器 xml_set_element_handler()函数

当 SAX 解析器遇到一个元素的开始标签或结束标签时，会调用起始或结束元素处理器。可以通过 xml_set_element_handler()函数来设置该处理器，语法格式如下所示：

```
xml_set_element_handler(parser, start_element, end_element)
```

该函数有 3 个参数，下面分别介绍这 3 个参数：第一个参数表示规定使用的 XML 解析器；第二个参数表示规定在元素开始时调用的函数；第三个参数表示规定在元素结束时调用的函数。第二个参数和第三个参数也可以是一个数组，其中包含对象引用和方法名，也可以说是

处理器函数的名称。

当 XML 解析器遇到元素起始标签时，调用起始元素处理器，并且规定该处理函数必须有 3 个参数，语法格式如下所示：

```
my_start_element_handler(parser, element, attributes)
```

这 3 个参数分别表示：调用处理程序的 XML 解析器的引用、起始元素的名称和解析器遇到的元素的属性数组，考虑到速度，该属性数组也是按引用传递。下面创建一个包含了元素起始处理器的示例，如代码 8.3 所示。

代码 8.3 起始元素处理函数

```
function start_element($inParser, $inName, &$inAttributes)
{
  $attributes = array( );
  foreach($inAttributes as $key)
  {
    $value = $inAttributes[$key];
    $attributes[] = "<font color=\"gray\">$key=\"$value\" </font>";
  }
  echo '&lt;<b>' . $inName . '</b> ' . join(' ', $attributes) . '&gt;';
}
```

当解析器到达元素结束标签时，就会调用结束元素处理器：

```
my_end_element_handler(parser, element);
```

它有两个参数：一个调用处理程序的 XML 解析器的引用和结束元素的名称。下面创建一个结束元素处理器的函数，如代码 8.4 所示。

代码 8.4 结束元素处理函数

```
function end_element($inParser, $inName)
{
  echo '&lt;<b>/$inName</b>&gt;';
}
```

2. 字符数据处理器 xml_set_character_data_handler()函数

元素之间的所有文本（例如字符数据、XML 术语中的 CDATA）由字符数据处理程序处理，即通过 xml_set_character_data_handler()函数设置的处理器会在遇到每一个字符数据块时被调用，该函数的语法格式如下所示：

```
xml_set_character_data_handler(parser, handler)
```

由上述语法格式知，字符数据处理器有两个参数：触发处理程序的 XML 解析器的引用和包含字符数据的字符串。下面创建一个简单的输出数据的字符数据处理器，具体代码如下所示：

```
function character_data($inParser, $inData)
{
  echo $inData;
}
```

3．处理指令 xml_set_ processing_instruction_handler()函数

在 XML 中处理指令（PI, Processing Instruction）用来将脚本或其他代码嵌入到文档中，PHP 代码本身就可被看成是一种遵循 XML 格式的处理指令，它以<?php…?>标签风格标记代码。XML 解析器在遇到处理指令时即调用相应的处理指令处理器。可以用 xml_set_processing_instruction_handler()函数设置处理器，该函数语法格式如下所示：

```
xml_set_processing_instruction_handler (parser, handler)
```

处理指令的格式如下：

```
<?target instructions ?>
```

处理指令处理器（processing instruction handler）有 3 个参数：触发处理器的解析器的引用、目标名称（如“php”）和处理指令。语法格式如下所示：

```
my_processing_instruction_handler(parser, target, instructions)
```

当解析文档时，用 eval()函数执行 PHP 代码。下面创建一个处理指令处理器，如代码 8.5 所示。

代码 8.5　处理指令处理器

```
function processing_instruction($inParser, $inTarget, $inCode)
{
  if ($inTarget === 'php')
  {
    eval($inCode);
  }
}
```

4．实体处理器

XML 中的占位符被称为实体，XML 提供了 5 种实体：&，>，<，"和'，并且 XML 文档还可以定义它们自己的实体。大多数实体定义不会触发事件，在调用其他处理器前，XML 解析器会对文档中的大多数实体进行扩展。

PHP 的 XML 库对外部实体和未解析实体提供特别支持。外部实体的替换文本是由文件名或者 URL 表示，而不是在 XML 文件中显式地给定外部实体的替换文本。读者可以定义一个处理器，并在字符数据中出现外部实体时调用，是否解析该文件或 URL 的内容，则由用户自身决定。未解析实体必须带有一个符号声明。程序员可以定义未解析实体和符号声明，并且未解析实体将在调用字符数据处理器前被删除，下面详细介绍这两种实体。

□ 外部实体

外部实体允许 XML 文档包含其他 XML 文档。一个外部实体引用的处理器打开被引用的文件，然后解析该文件，并且将结果包含在当前文档中。可用 xml_set_external_entity_ref_handler()函数来设置处理器，该函数有两个参数：XML 解析器的引用和处理器函数的名称。如果处理器被成功建立，该函数返回 true，否则返回 false。该函数的语法格式如下所示：

```
bool xml_set_external_entity_ref_handler(parser, handler)
```

外部实体处理器规定必须有 5 个参数：触发处理器的 XML 解析器、实体名称、分解实体标识符的基本 URL（当前为空）、系统标识符（例如文件名）和实体的公共标识符。如果外部实体处理器返回一个 false 值（或没有返回任何值），XML 解析器将停止并报告一个 XML_ERROR_EXTERNAL_ENTITY_HANDLING 错误。如果返回 true，则继续解析。下面创建一个示例，该示例所用到的 XML 文档如代码 8.6 所示。

代码 8.6　XML 文档

```
<?xml version="1.0" encoding="gb2312"?>
<note>
<date>2008 年 11 月 14 日 星期五&amp;</date>
<weather>小雨&amp;</weather>
<heading>今天学习 XML 实体</heading>
<body>&amp;, &gt;, &lt;, &quot; 和 &apos;</body>
</note>
```

该示例介绍了如何解析外部引用的 XML 文档。在示例中定义了两个函数来创建 XML 解析器和进行解析工作，可以用其来解析顶层文档和由外部引用包含的任何文档，外部实体引用处理器只是简单地确定哪些文件要传递给这些函数，如代码 8.7 所示。

代码 8.7　外部实体

```
<?php
$parser=xml_parser_create();//===创建一个解析器
function char($parser,$data)
  {
  echo $data;
  }
function ext_ent_handler($parser,$ent,$base,$sysID,$pubID)
  {
  echo "$ent";
  echo "$sysID";
  echo "$pubID";
  }
xml_set_character_data_handler($parser,"char");
xml_set_external_entity_ref_handler($parser, "ext_ent_handler");
$fp=fopen("test.xml","r");
while ($data=fread($fp,4096))
```

```
  {
  xml_parse($parser,$data,feof($fp)) or
  die (sprintf("XML Error: %s at line %d",
  xml_error_string(xml_get_error_code($parser)),
  xml_get_current_line_number($parser)));
  }
xml_parser_free($parser);
?>
```

该段代码执行结果如下所示：

```
2008 年 11 月 14 日 星期五& 小雨& 今天学习 XML 实体 &, >, <, " 和 '
```

□ **未解析实体**

未解析实体的声明要带有一个符号声明：

```
<!DOCTYPE doc [
 <!NOTATION jpeg SYSTEM "image/jpeg">
 <!ENTITY logo SYSTEM "php-tiny.jpg" NDATA jpeg>
]>
```

当解析器在 XML 文档中遇到符号声明时，调用 xml_set_notation_decl_handler()函数，使用该函数可以注册一个符号声明处理器。xml_set_notation_decl_handler()函数有两个参数：XML 解析器和符号处理器。如果处理器被成功建立返回 true，否则返回 false。

5. 默认处理器 xml_set_default_handler()函数

只要解析器在 XML 文件中找到数据，都可以调用 xml_set_default_handler()函数。xml_set_default_handler()函数为 XML 解析器建立默认的数据处理器，如果处理器被成功创建，该函数将返回 true，否则返回 false，语法格式如下所示：

```
xml_set_default_handler(parser, handler)
```

由 xml_set_default_handler()函数语法格式知，该函数有两个参数，其中第一个参数表示要使用的 XML 解析器；handler 参数表示事件处理器使用的函数。下面使用该函数创建一个示例如代码 8.8 所示，该示例所使用 XML 文件如代码 8.8 所示。

代码 8.8　使用默认处理器

```
<?php
$parser=xml_parser_create();
function mydefault($parser,$data)
{
   echo $data;
}
xml_set_default_handler($parser,"mydefault");
$fp=fopen("test.xml","r");
while ($data=fread($fp,4096))
```

```
{
  xml_parse($parser,$data,feof($fp)) or
  die (sprintf("XML Error: %s at line %d",
  xml_error_string(xml_get_error_code($parser)),
  xml_get_current_line_number($parser)));
}
xml_parser_free($parser);
?>
```

代码 8.8 执行结果如下所示：

```
2008 年 11 月 14 日 星期五& 小雨& 今天学习 XML 实体 &, >, <, " 和 &apos;
```

在浏览器中，查看该页面的源文件，如代码 8.9 所示。

代码 8.9　页面源代码

```
<note>
<date>2008 年 11 月 14 日 星期五&amp;</date>
<weather>小雨&amp;</weather>
<heading>今天学习 XML 实体</heading>
<body>&amp;, &gt;, &lt;, &quot; 和 &apos;</body>
</note>
```

6. 选项

XML 解析器使用 xml_parser_set_option()函数可以控制源文档编码、目标文档编码和大小写形式。使用 xml_parser_get_option()函数可得到 XML 解析器的选项设置信息，下面介绍 XML 文档所支持的选项信息。

- **字符编码**

PHP 使用的 XML 解析器支持各种不同字符编码的 Unicode 数据。在 XML 文档内部，PHP 字符串通常用 UTF-8 编码，但是 XML 解析器解析的文档可以是 ISO-8859-1、US-ASCII 或 UTF-8。在创建 XML 解析器时，可以指定要解析的文件的编码。如果省略了，则假定源文件按 ISO-8859-1 编码。如果源文件中的字符超出了其编码范围，XML 解析器将返回一个错误，并立即停止处理文档。

XML 解析器的目标编码是将数据传递给处理器程序函数时的编码。通常它与源文件的编码一致。在 XML 解析器生存期的任何时候，目标编码都可以被改变。任何超出编码范围的字符都被问号（?）所取代。

使用常量 XML_OPTION_TARGET_ENCODING 可获得或者设置传递给回调函数的文本编码，其允许值为“ISO-8859-1”（默认值）、“US-ASCII”和“UTF-8”。

- **大小写**

XML 文档中的所有元素和属性名称都会被转换成大写形式。可通过将 XML_OPTION_CASE_FOLDING 选项设为 false 来禁止这种转换，即使用区分大小写的名称。

```
xml_parser_set_option(XML_OPTION_CASE_FOLDING, false);
```

7．使用 XML 解析器 xml_parser_create()函数

在 PHP 中使用 xml_parser_create()函数创建 XML 解析器，然后设置该解析器的处理器和选项，再用 xml_parse()函数将数据传递给解析器处理，直到数据处理完毕或者解析器返回一个错误。一旦处理完毕，调用 xml_parser_free()函数释放解析器。xml_parser_create()返回一个 XML 解析器，使用 xml_parser_create()函数创建 XML 解析器的语法格式如下所示：

```
$parser = xml_parser_create([encoding]);
```

由 xml_parser_create()函数语法格式知，该函数有一个可选参数，表示指定将要解析的文本的编码，例如 ISO-8859-1、US-ASCII 或 UTF-8。xml_parse()函数如果解析成功会返回 true，失败则返回 false。xml_parse()函数语法格式如下所示：

```
$success = xml_parse(parser, data[, final ]);
```

data 参数是要处理的 XML 字符串，为了使最后一块数据被解析，可选的 final 参数应设为 true。为了使处理嵌套文档变得简单，可以编写一个函数来创建解析器并设置其选项和处理器。该函数将选项和处理器设置放在同一个地方，而不是在外部实体引用的处理器中复制它们。

8.2.3　使用 DOM 库对 XML 文档解析

在 PHP 中 DOM 也可以看作是一组 API（Application Program Interface，应用程序接口），它将 HTML 文档和 XML 文档看成一个文档对象，在里面存放的是对这些文档操作的属性和方法的定义。如果编程语言实现了这些属性和方法，就可以对文档对象中的数据进行存取，并且利用程序对数据进行进一步的处理。

PHP 语言实现了 DOM 接口，这就意味着可以利用 W3C 提供的 DOM 接口操作 XML 文档。在使用 DOM 接口对 XML 文档操作之前，需要将 XML 文档加载到内存中，形成一棵节点树。树上的每一个节点，就是 XML 文档的每一个标记和相应的标记内容。使用 DOM 接口对 XML 进行操作，在 PHP 中可以使用两个类库，一个为 DOM Functions 函数库，另外一个为 DOM XML Functions 函数库。可以调用第一个函数库中的函数，这是因为其为内置函数库；使用第二个函数库需要在 php.ini 文件中引入。本节主要介绍使用内置函数库对 XML 文件解析，在此就不再介绍如何在 php.ini 文件中引入 DOM XMLFunctions 函数库。下面结合对 XML 文件的操作，介绍 DOM Functions 函数库中的函数。

1．遍历节点

下面编写一个使用 DOM Functions 函数库中的函数从 XML 文件中获取数据的示例。在编写该示例之前，首先介绍示例中运用到的函数库的函数，这些函数如表 8-1 所示。

表 8-1　示例中需要运用的函数说明

函数名称	函数说明
DOMDocument()	创建一个 DOM 对象
load()	加载一个指定 XML 文件到内存
getElementsByTagName()	获取一个指定节点的节点对象
item()	获取指定索引值的节点对象
NodeValue()	节点值

通过表 8-1 对函数的学习，现在就可以创建一个演示遍历 XML 文档的数据的示例了。首先在项目中创建一个 book.xml 文档，即在 MyWeb\Apache\htdocs 目录下创建，该 XML 文档主要存储一些书籍相关的信息，内容如代码 8.10 所示。

代码 8.10　XML 文档内容

```
<?xml version="1.0" encoding="gb2312"?>
<bookstore>
  <book id="0">
    <title>PHP 程序设计</title>
    <author sex="男">唐晓阳</author>
    <myyear>2007 年 2 月</myyear>
    <price>79</price>
  </book>
  <book>
    <title>PHP 程序开发</title>
    <author sex="女">裴亚敏</author>
    <myyear>2008 年 11 月</myyear>
    <price>65</price>
  </book>
</bookstore>
```

接下来编写 PHP 脚本代码，从而读取 XML 文档中的内容。在 PHP 程序中首先要创建一个 DOMDocument 对象，然后使用该对象加载 XML 文件，最后读取文档中的内容，如代码 8.11 所示。

代码 8.11　PHP 脚本程序

```
<?php
  $doc = new DOMDocument();
  $doc->load( 'book.xml' );
  $bookstore = $doc->getElementsByTagName( "book" );
  foreach( $bookstore as $book )
  {
    $titles = $book->getElementsByTagName( "title" );
    $title =$titles->item(0)->nodeValue;
    $authors = $book->getElementsByTagName( "author" );
```

```
        $author = $authors->item(0)->nodeValue;
        $years = $book->getElementsByTagName( "myyear" );
        $year = $years->item(0)->nodeValue;
        $prices =$book->getElementsByTagName("price");
        $price=$prices->item(0)->nodeValue;
        echo "Title->$title - Author->$author -
             Time->$year - Price->$price \n<br/>";
      }
    ?>
```

保存代码 8.11，文件名称为 Demo-XMLBook.php，并且将该文件与 book.xml 文件保存在同一个目录中。打开 IE 浏览器，在地址栏中输入 http://localhost/Demo-XMLBook.php，单击【转到】按钮，会显示如图 8-1 所示的窗口。

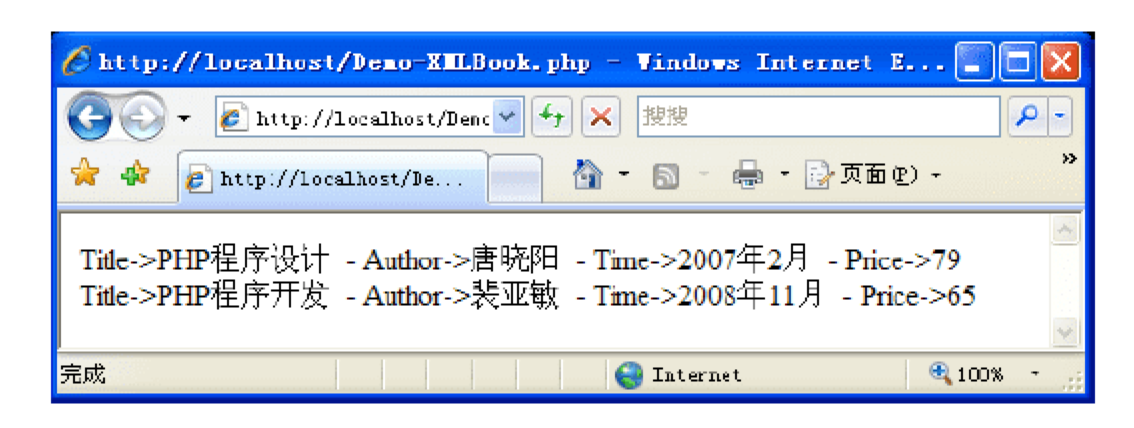


图 8-1　读取 XML 文档

在代码 8.11 中，首先创建了一个 DOM 对象，然后加载一个指定的 XML 文件到内存中，并形成一个节点树。然后使用语句“$doc->getElementsByTagName("book")”创建了一个节点集对象$bookstore，这时该对象就指向了节点树中节点名称为 book 的所有节点及其子节点的集合。

如果想在页面上显示 XML 文档中的内容，也可以不遍历各个节点，直接显示到页面上，做到这点很容易，只需要使用 saveXML()函数即可，具体代码如下所示：

```
   $doc = new DOMDocument();
   $doc->load( 'book.xml' );
   print $doc->saveXML();
```

2．增加节点

在 PHP 中使用 createAttribute()函数和 createTextNode()函数，可以向 XML 文档中添加新的属性或新的节点，createAttribute()函数表示创建一个属性；createTextNode()表示向 XML 文档中创建一个新的节点。下面创建一个示例，具体讲解使用 PHP 向 XML 文档中添加节点或属性，该示例使用的 XML 文档如代码 8.12 所示，该 XML 文档文件名为“index.xml”，该示例如代码 8.13 所示。

代码 8.12　XML 文档代码

```
<?xml version="1.0" encoding="GB2312"?>
<books>
```

```
<book id="1">
<author sex="男">唐晓阳</author>
<publisher>人民邮电出版社</publisher>
<title>PHP</title>
</book>
<book>
<author sex="女">郏东方</author>
<publisher>清华大学出版社</publisher>
<title>C#</title>
</book>
</books>
```

代码 8.13　添加节点/属性

```
<?php
$doc = new DOMDocument("1.0");
$doc->load("index.xml");
//创建节点 book
$node =$doc->createElement("book");
//新建节点 book 的属性 id
$id=$doc->createAttribute("id");
$idText=$doc->createTextNode("3");
$id->appendChild($idText);
$node->appendChild($id);
//创建 book 节点的子节点 author
$author=$doc->createElement("author");
$author=$node->appendChild($author);
$authorText=$doc->createTextNode("df");
$author->appendChild($authorText);
//创建 book 节点的子节点 title
$title=$doc->createElement("title");
$title=$node->appendChild($title);
$titleText=$doc->createTextNode("C++");
$title->appendChild($titleText);
//将 book 节点添加到 XML 文件中
$doc->getElementsByTagName('book')->item(0)->appendChild($node);
$doc->save("index.xml");
echo "<hr/><a href=\"index.xml\">查看 XML 文档</a>";
?>
```

在代码 8.13 中，使用语句$doc->createElement("book")创建一个名称为 book 的节点对象 node，$id=$doc->createAttribute("id")语句表示创建属性对象 id，并使用$idText=$doc->createTextNode("3")语句创建了一个内容为 3 的文本节点对象 idText。接下来将节点对象 idText 追加到节点对象 id，再将属性对象 id 追加到节点对象 node 上。如果要创建 node 的子节点，其过程与创建 node 的属性基本一样，只不过需要使用方法 createElement()而不是 create-

Attribute()。最后，不要忘了使用 appendchild()方法将 node 节点对象添加到 XML 文档中。最后在浏览器中查看 index.xml 文档，页面效果如图 8-2 所示。

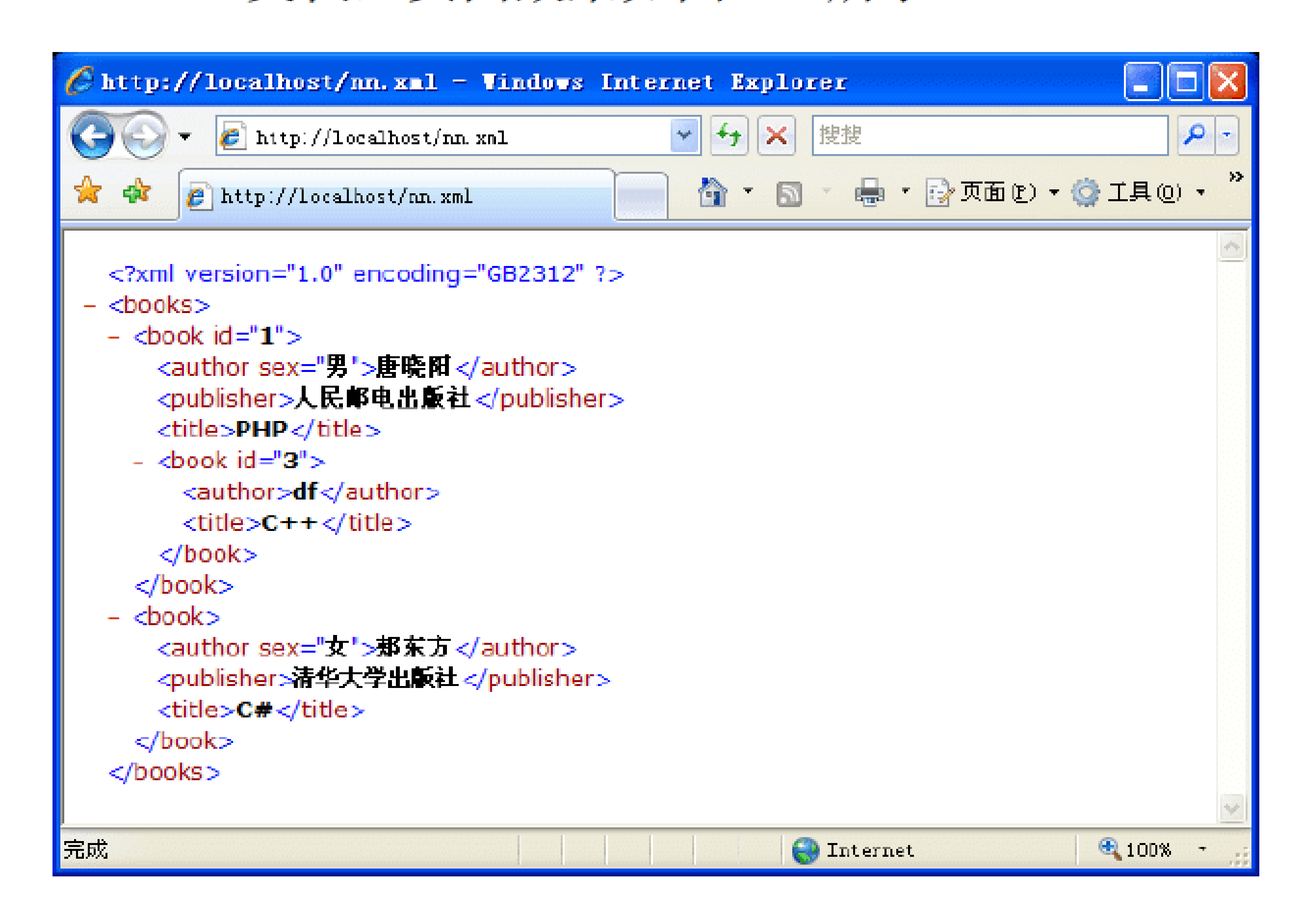


图 8-2　XML 文档内容

3．修改 XML 文档节点

使用 DOM 可以修改 XML 文档中的节点或者属性，修改这些节点其实十分简单，只需要使用一些函数就可以实现，这些函数如表 8-2 所示。

表 8-2　修改节点函数

函数名称	函数说明
domxpath($dom)	获取一个搜索对象
query("/books/book/title")	获取指定路径的节点对象
createElement("title")	创建一个名称为 title 的节点
appendChild("name")	追加一个内容为 name 的子节点
replaceChild($newTitle, $firstTitle)	用 newtitle 节点替换 firstTitle 节点
setAttribute("sex", "aaa")	设置属性 sex 的值
save("newfile.xml")	将节点树保存为 newfile.xml 文件
getAttribute("title")	获取属性 title 的值

下面创建一个示例，具体讲解如何使用这些函数修改 XML 文档中的节点和节点属性，该示例使用的是 index.xml 文件，该示例的 PHP 脚本程序如代码 8.14 所示。

代码 8.14　修改 XML 文档中节点和节点属性

```
<?php
    $dom = new DomDocument("1.0");
    $dom->load("index.xml");
    $xp = new domxpath($dom);
    $firstTitle = $xp->query("/books/book/title")->item(0);
    $newTitle = $dom->createElement("title");
```

```
    $newTitle->appendChild(new DOMText("My New Text!!!"));
    $firstTitle->parentNode->replaceChild($newTitle, $firstTitle);
    //修改属性
    $firstTitle1 = $xp->query("/books/book/author")->item(0);
    $firstTitle1->setAttribute("sex", "123");
    $dom->save("Mynewfile.xml");
    echo $firstTitle1->getAttribute("title");
    echo "<hr/><a href=\"Mynewfile.xml\">查看新 XML 文档</a>";
?>
```

在代码 8.14 中，首先创建了一个 DOM 对象，然后将 index.xml 文件加载到内存中，再用语句“new domxpath($dom)”创建了一个查询对象$xp，该对象可以获取节点树任意位置的节点。在下面使用方法 query 获取“/books/book/title”路径下的节点对象，语句 createElement("title")表示创建一个新的节点，节点名称为 title。使用 appendChild()为新的节点添加一个文本节点，文本节点是标记之间的内容。在下面使用 replaceChild()用新建的节点 newTitle 替换旧的节点 firstTitle。这段代码表示修改节点的内容。最后保存该文件，并重命名为 Demo-Index.php，打开 IE 浏览器，转到该页面，在该页面上会显示超链接“查看新 XML 文档”，单击该超链接，转到显示 XML 文档的页面，效果如图 8-3 所示。



图 8-3　显示修改后 XML 文档内容页面

4．删除节点

在 PHP 中删除 XML 文档中的指定节点或属性，只需要使用 removeChild()函数和 removeAttribute()函数。下面创建一个示例，具体说明如何使用这两个函数删除 XML 文档中的节点或属性，如代码 8.15 所示。

代码 8.15　删除 XML 文档中的节点和属性

```
<?php
    $dom = new DomDocument("1.0");
    $dom->load("index.xml");
```

```
    $xp = new domxpath($dom);
//获取符合指定路径，并且该节点的属性 id 的值为 1 的节点对象$titles
    $titles = $xp->query("/books/book[@id = \"1\"]");
    $dom->documentElement->removeChild($titles->item(0));
    $dom->save("Delete.xml");
   echo "<hr/><a href=\"Delete.xml\">查看删除节点后的 XML 文档</a>";
 ?>
```

在代码 8.15 中，创建了一个查询对象$xp，然后获取指定路径和指定属性值的节点，最后直接使用$dom->documentElement->removeChild($titles->item(0))移除指定节点。

8.2.4　使用 SimpleXML 处理 XML

在 PHP 中，可以使用 PHP 提供的 SimpleXML 类库，处理非常简单的 XML 文档，不过 SimpleXML 无法像 DOM 扩展那样生成 XML 文档，也不像 Expat 扩展那样灵活可扩展且高效，但是用其来解析和使用 XML 文档非常容易。

SimpleXML 可以读取一个 XML 文件、字符串和 DOM 文档（由 DOM 扩展生成），然后生成一个对象，该对象的属性提供每个节点元素的访问途径，并且可以使用数字索引，或非数字索引来访问属性，还可以对任何从某一项中获取的文本值进行字符串转换。下面使用 SimpleXML 创建一个示例，该示例所使用的 XML 文档为 books.xml 文档，内容如代码 8.16 所示。该示例使用 SimpleXML 获取 XML 文档内容，PHP 脚本程序如代码 8.17 所示。

代码 8.16　XML 文档内容

```
<?xml version="1.0" encoding="GB2312"?>
<books>
   <book>
     <booktitle>PHP 程序设计</booktitle>
     <author>Yound Tang</author>
     <price>38</price>
   </book>
   <book>
    <booktitle>C#程序设计</booktitle>
     <author>YaLan Pei</author>
     <price>38</price>
   </book>
   <book>
     <booktitle>Java 程序设计</booktitle>
     <author>East Jia</author>
     <price>38</price>
   </book>
</books>
```

代码 8.17　显示 XML 文档内容

```
<?php
```

```
if(file_exists('books.xml'))
{
    $xml = simplexml_load_file('books.xml');
    }
    echo $xml->asXML();
?>
```

在代码 8.17 中，首先使用 file_exists()判断 XML 文档是否存在，然后使用 simplexml_load_file()加载 XML 文档，最后使用函数 asXML()以字符串形式返回 XML 文档。如果此时查看页面源文件，会发现源文件内容为 XML 文档格式的数据，和代码 8.16 相同。

由示例代码 8.17 知，在 PHP 中使用 SimpleXML 读取 XML 文档非常容易，这主要是由于 SimpleXML 支持 asXML()函数，像这样的函数 SimpleXML 支持的还很多，下面详细介绍这些函数。

1. children()函数

该函数主要获取指定节点的子结点，语法格式如下所示：

```
class SimpleXMLElement
{
string children(ns,is_prefix)
}
```

下面创建一个示例，具体讲解如何运用该函数。示例中运用的 XML 文档如代码 8.18 所示，本节所有示例使用的 XML 文档均为该文档，以下列举的示例将不再说明。该示例如代码 8.19 所示。

代码 8.18　mybook.xml 文档内容

```
<?xml version="1.0" encoding="GB2312"?>
  <book>
    <booktitle>PHP 程序设计</booktitle>
    <author>唐晓阳</author>
    <price>68</price>
  </book>
```

代码 8.19　使用 children()函数

```
<?php
if(file_exists('mybook.xml'))
{
  $xml = simplexml_load_file("mybook.xml");
  foreach ($xml->children() as $child)
  {
    echo "Children Node:" . $child."<br/>";
  }
```

```
}
?>
```

保存代码 8.19，并重命名为“Demo-children.php”，执行该段代码，结果如下所示：

```
Children Node:PHP 程序设计
Children Node:唐晓阳
Children Node:68
```

2. addChild()函数

addChild()函数向指定的 XML 节点添加节点，该函数返回一个 SimpleXMLElement 对象，这个对象表示添加到 XML 节点的子元素，该函数语法格式如下所示：

```
string addChild(name,value,ns)
```

由 addChild()函数语法格式知，该函数有 3 个参数，分别表示要添加的子元素名称、子元素值和可选参数子元素命名空间。下面使用该函数创建一个示例，如代码 8.20 所示。

代码 8.20　使用 addChild()函数

```
<?php
if(file exists('mybook.xml'))
{
   $xml = simplexml_load_file("mybook.xml");
}
$xml->author[0]->addChild("secondAuthor", "裴亚敏");
foreach ($xml->author->children() as $child)
{
   echo $child;
}
?>
```

保存并执行该段代码，结果如下所示：

```
裴亚敏
```

3. attributes()函数

该函数主要用于获取 SimpleXML 元素属性，该函数提供一个 XML 标签中定义的属性和值，语法格式如下所示：

```
string attributes(ns,is_prefix)
```

由该函数语法格式知，有两个可选参数，第一个参数表示被检索的属性的命名空间，第二个参数默认值为 false。在创建该示例之前，首先修改 XML 文档，为 author 节点添加一个“sex="男"”的属性，然后编写 PHP 脚本程序，如代码 8.21 所示。

代码 8.21　使用 attributes()函数

```
<?php
  $xml = simplexml_load_file("mybook.xml");
  foreach($xml->author[0]->attributes() as $a => $b)
  {
      echo $a,'="',$b,'"';
  }
?>
```

保存并执行该段代码，结果如下所示：

```
sex="男"
```

4．addAttribute()函数

addAttribute()函数主要是为 SimpleXML 元素添加一个属性，并且无返回值。函数语法格式如下所示：

```
string addAttribute(name,value,ns)
```

由该函数语法格式知，函数有 3 个参数：属性名称、属性值和属性命名空间，其中第三个参数为可选参数。下面使用该函数创建一个示例，如代码 8.22 所示。

代码 8.22　使用 addAttribute()函数

```
 <?php
  $xml = simplexml_load_file("mybook.xml");
  $xml->author[0]->addAttribute("age", "25");
  foreach($xml->author[0]->attributes() as $a => $b)
  {
     echo $a,'="',$b,'"';
     echo "<br/>";
  }
?>
```

保存并执行代码 8.22，结果如下所示：

```
sex="男"
age="25"
```

5．getName()函数

getName()函数从 SimpleXMLElement 对象获取 XML 元素的名称，如果执行成功返回当前的 XML 元素名称，否则返回 false。该函数语法格式如下所示：

```
string getName()
```

下面创建一个示例，具体讲解如何运用该函数，如代码 8.23 所示。

代码 8.23　使用 getName()函数

```
<?php
  if (file_exists('mybook.xml'))
  {
     $xml = simplexml_load_file('mybook.xml');
  }
  echo $xml->getName();
  echo"<br/>";
  foreach($xml->children() as $child)
  {
     echo $child->getName();
     echo "<br/>";
  }
?>
```

保存并执行该段代码，结果如下所示：

```
Book
Booktitle
Author
Price
```

6．xpath()函数

XPath 是一个 W3C 标准，提供了标记 XML 节点的一种基于路径的语法，相当直观。例如引用 mybook.xml 文档，可以使用表达式 book/author 获取所有 author 节点。XPath 还提供了一组函数，可以根据值有选择性地获取节点。下面创建一个使用该方法的示例，如代码 8.24 所示。

代码 8.24　使用 xpath()函数

```
<?php
   $xml=simplexml_load_file("mybook.xml");
   $books=$xml->xpath("/book/author");
   foreach($books as $book)
   {
       echo "$book<br/>";
   }
?>
```

保存并执行该段代码，结果如下所示：

```
唐晓阳
```

7．registerXPathNamespace()函数

registerXPathNamespace()函数为下一次 XPath 查询创建命名空间语境，该函数语法格式如

下所示：

```
string registerXPathNamespace(prefix,ns)
```

由语法格式知，该函数有两个参数，其中第一个参数表示命名空间的前缀，第二个参数表示必须匹配 XML 文档中的命名空间。下面使用该函数创建一个示例，PHP 脚本程序如下所示：

```
if (file_exists('mybook.xml'))
{
   $xml = simplexml_load_file('mybook.xml');
}
$xml->registerXPathNamespace("m", "http://www.itzcn.com");
$result = $xml->xpath("m:author");
foreach ($result as $message)
{
   echo $message;
}
```

保存并执行该段代码，结果如下所示：

```
唐晓阳
```

8. getDocNamespaces()函数和 getNamespace()函数

getDocNamespaces()函数从 SimpleXMLElement 对象返回在 XML 文档中声明的命名空间。如果成功，返回包含命名空间名称的数组，否则返回 false。该函数语法格式如下所示：

```
string getDocNamespaces(recursive)
```

由语法格式知，该函数有一个可选参数，表示是否返回父子节点中的所有命名空间，默认为 false。下面创建一个示例，具体讲解如何运用 getDocNamespaces()函数获取命名空间，在编写 PHP 脚本程序之前，首先更改 mybook.xml 文档，修改后的文档内容如代码 8.25 所示，该示例脚本程序如代码 8.26 所示。

代码 8.25　mybook.xml 文档内容

```
<?xml version="1.0" encoding="GB2312"?>
  <book xmlns:b="http://www.ITZCN.com">
    <booktitle>PHP 程序设计</booktitle>
    <author sex="男">唐晓阳</author>
    <price>68</price>
  </book>
```

代码 8.26　使用 getDocNamespaces()函数

```
<?
  if (file_exists('mybook.xml'))
```

```
    {
      $xml = simplexml_load_file('mybook.xml');
    }
    print_r($xml->getDocNamespaces());
?>
```


保存代码 8.26 并执行，结果如下所示：

```
Array (
  [b] => http://www.ITZCN.com
)
```

SimpleXML 还支持一个同样获取命名空间的 getNamespace()函数，不过该函数获取在 XML 文档中使用的命名空间，语法格式如下所示：

```
string getNamespace(recursive)
```

下面使用 getNamespace()创建一个示例，运用的 XML 文档为代码 8.25 所示 XML 文档内容，该示例 PHP 脚本程序如代码 8.27 所示。

代码 8.27　运用 getNamespace()函数

```
    if (file_exists('mybook.xml'))
    {
      $xml = simplexml_load_file('mybook.xml');
    }
    print_r($xml->getNamespace());
```

保存并执行该段代码，结果如下所示：

```
Array ( )
```

9. simplexml_load_string()函数

simplexml_load_string()函数将 XML 字符串载入对象中，如果失败则返回 false。该函数的语法格式如下所示：

```
simplexml_load_file(string,class,options,ns,is_prefix)
```

由语法格式知，该函数有 5 个参数，如表 8-3 所示。

表 8-3　simplexml_load_string()函数参数说明

参数名称	参数说明
string	该参数是必需参数，表示要使用的 XML 字符串
class	该参数为可选参数，新对象的 class
options	该参数为可选参数，附加的 Libxml 参数
ns	该参数为可选参数
prefix	该参数为可选参数

下面使用该函数创建一个示例，如代码 8.28 所示。

代码 8.28　使用 simplexml_load_string()函数

```
<?php
$xmlstring = <<<XML
<?xml version="1.0" encoding="GB2312"?>
  <book>
    <booktitle>PHP 程序设计</booktitle>
    <author sex="男">唐晓阳</author>
    <price>68</price>
  </book>
XML;
$xml = simplexml_load_string($xmlstring);
var_dump($xml);
?>
```

保存代码 8.28 并执行，结果如下所示：

```
object(SimpleXMLElement)#1 (3)
{
    ["booktitle"]=> string(15) "PHP 程序设计"
    ["author"]=> string(9) "唐晓阳"
    ["price"]=> string(2) "68"
}
```

10．simplexml_import_dom()函数

simplexml_import_dom()函数将 DOM 节点转换为 simpleXMLElement 对象，如果失败该函数返回 false，语法格式如下所示：

```
simplexml_import_dom(data,class)
```

该函数有两个参数，data 表示要使用的 DOM 节点，为必选参数。class 表示指定新对象的类，为可选参数。下面使用该函数创建一个示例，如代码 8.29 所示。

代码 8.29　使用 simplexml_import_dom()函数

```
<?php
$dom = new domDocument;
$dom->loadXML('<book><author>Yound</author></book>');
$xml = simplexml_import_dom($dom);
echo $xml->author;
?>
```

保存代码 8.29 并执行，结果如下所示：

```
Yound
```

8.3　客户端处理 XML

在服务器端，可以使用 PHP 脚本程序处理 XML 文件，但这样会占据服务器线程处理程序，加大了服务器的工作量，减慢服务器的执行速度。有时可以在客户端处理 XML 文件信息，这通常都是采用 JavaScript 脚本程序实现。本节主要讲解如何使用脚本在客户端读取 XML 文件信息。

JavaScript 是一种基于对象和事件驱动，并具有较高安全性的脚本语言。JavaScript 既可以用到客户端又可以用到服务器端，服务器端和客户端 JavaScript 共享相同的核心语言。使用 JavaScript 解析 XML 文档，可以进行遍历、添加、删除和修改等操作。下面创建一个示例，具体讲解如何使用 JavaScript 获取节点和相应的值，本示例使用的 XML 文档为 index.xml 文件，该文件信息如代码 8.30 所示，该示例的 JavaScript 脚本程序如代码 8.31 所示。

代码 8.30　index.xml 文件内容

```
<?xml version="1.0" encoding="GB2312"?>
<books>
<book id="1">
<author sex="男">唐晓阳</author>
<publisher>人民邮电出版社</publisher>
<title>PHP 程序设计</title>
</book>
<book>
<author sex="男">郑东方</author>
<publisher>清华大学出版社</publisher>
<title>C#程序设计</title>
</book>
</books>
```

代码 8.31　使用 JavaScript 脚本处理 XML 文件

```
<script language="JavaScript">
  var xmlDocument = new ActiveXObject("Microsoft.XMLDOM");
  xmlDocument.load("index.xml");
   var obj=xmlDocument.documentElement.childNodes;
    for(var i=0;i<obj.length;i++)
    {
       document.write(obj.item(i).childNodes.item(0).nodeName+":  ");
       document.write(obj.item(i).childNodes.item(0).text+"    ");
       document.write(obj.item(i).childNodes.item(1).nodeName+":  ");
       document.write(obj.item(i).childNodes.item(1).text+"    ");
       document.write(obj.item(i).childNodes.item(2).nodeName+":  ");
       document.write(obj.item(i).childNodes.item(2).text+"<br/>");
```

```
    }
</script>
```

将上述代码保存，命名为 demo-javascript.html，并且该文件要和 index.xml 文件保存在同一个目录下，打开 demo-javascript.html 文件，执行结果如图 8-4 所示。

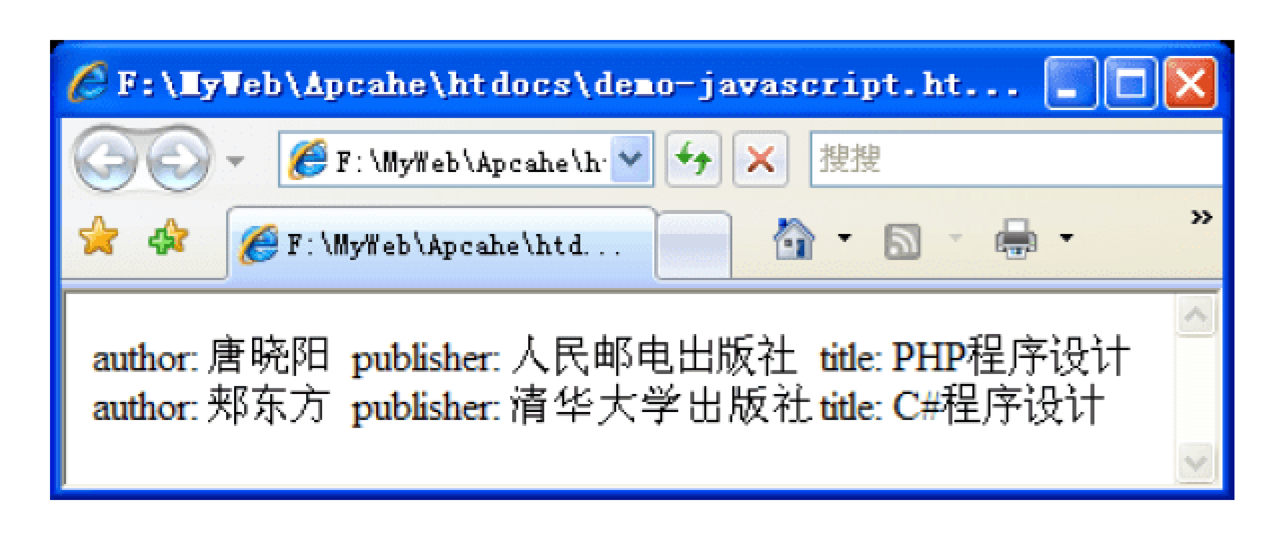


图 8-4 显示 XML 节点

在代码 8.31 中，语句 new ActiveXObject("Microsoft.XMLDOM")表示创建一个 DOM 对象，然后使用函数 load()加载指定的 XML 文件到内存中进行操作，xmlDocument.documentElement.childNodes 语句表示创建根节点下的所有节点的集合。其中 childNodes 表示子节点的集合；item()方法表示依据节点具有的子节点的索引值返回该子节点；属性 nodeName 表示节点的名称，即标记的名称；属性 text 表示标记的内容。

第 3 篇　Ajax 基础篇

第 9 章 Ajax 概述



Ajax 是 Web 2.0 的核心，全称是 Asynchronous JavaScript And XML（异步 JavaScript 和 XML），它不是一项新技术，而是很多成熟技术的集合。基于 Ajax 的开发与传统 Web 开发模式最大的区别就在于传输数据的方式不同，前者为异步，后者为同步。Ajax 的实现在性能上有很大提高，用户的行为经过 Ajax 引擎的处理，使得客户端可以只获取需要的数据。服务器端的组织形式将功能划分得更细，服务器只为有用的数据进行工作，提高运行效率。

本章首先对 Ajax 进行全面的介绍，例如 Ajax 的相关概念和技术，并对 Ajax 运行机制进行简单的概述。接着介绍传统 Web 应用解决方案与 Ajax 应用的区别以及 Ajax 的优势。最后通过实例具体演示 Ajax 的使用。



- 了解 Web 2.0 的概念
- 理解什么是 Ajax
- 掌握 Ajax 相关技术和核心内容
- 理解 Ajax 的工作原理和基本原则
- 熟悉 Ajax 实例开发过程

9.1 Web 2.0 与 Ajax 简介

Web 2.0 是相对 Web 1.0 的新一类互联网应用的统称。由 Web 1.0 通过网络浏览器浏览 HTML 网页模式转向内容更丰富，联系性和工具性更强的 Web 2.0 互联网模式。从基本构成单元上，是由“网页”向“发表/记录的信息”发展。从工具上，是由互联网浏览器向各类浏览器和 RSS 阅读器等内容发展。运行机制上，由 Client Server 向 Web Services 转变。

Ajax 可以构建动态和响应更灵敏的 Web 应用程序，关键在于对浏览器端的 JavaScript、XHTML 和与服务器异步通信的组合。Ajax 使用支持以上技术的 Web 浏览器作为运行平台。这些浏览器目前包括：Mozilla、Firefox、Internet Explorer、Opera、Konqueror 及 Safari。

9.1.1 Web 2.0 简介

Web 2.0 是指网络泡沫之后，使用 Web 方式的巨大变化。Web 2.0 不是一个技术性的概念，

只是标识现今互联网已经发展到一个新阶段，以至于出现和 1.0 时代质变的内容。作为一个流行概念，Web 2.0 揭示互联网发展趋势。由于设备普及、资费下降、观念演进，使得网民的数量大量提高，网民对互联网的应用也在向前所未有的阶段迈进，这正是 Web 2.0 的基础。网民的参与和自我表达越来越强，这使得互联网上的内容和个人页面越来越多；并且网络与现实世界越来越密切相关，这一切使得 SNS（社会关系网络服务）和信息整理、信息处理（如搜索服务）变得尤为重要。

9.1.2 什么是 Ajax

Ajax 是异步 JavaScript 和 XML 的缩写，这只是最初的观点，由 Jesse James Garrett 创造出来。但是，Ajax 现在的覆盖面已经有了进一步的扩展，允许浏览器与服务器通信而无需刷新当前页面的技术都包含在其中。

Ajax 并不是一种全新的技术，而更像一种技巧，是将过去的几种技术巧妙结合的技巧。真正与 Ajax 相关的新名词应该是 XMLHttpRequest，最早在 Internet Explorer 5 中出现，最近开始在多数浏览器得到支持，是用于实现异步通信的对象。正如前面所说，B/S 模式是利用浏览器作为其通用的客户端，所以要想异步通讯成为可能，必须要得到浏览器的支持。如果不是浏览器对 XMLHttpRequest 对象的广泛支持，可能不会看到 Ajax 的今天，更不会看到许多对 Ajax 的著名应用，如 Google Map，Google Suggest 和 Ta-da List 等。

Ajax 的处理过程由事件触发，创建一个 XMLHttpRequest 对象，将 HTTP 方法（GET/POST）和目标 url 以及请求返回后的回调函数设置到 XMLHttpRequest 对象，通过 XMLHttpRequest 向服务器发送请求，请求发送后继续响应用户的界面交互，只有等到请求真正从服务器返回时，才调用 callback()函数，对响应数据进行处理。

9.1.3 Ajax 运行机制

Ajax 是在 JavaScript 中利用 XMLHttpRequest 对象直接与服务器进行异步通信，以更新页面的局部信息，而不用刷新整个页面的机制。由于它是局部更新，所以速度更快；由于利用现有的、标准的数据交互协议（HTTP 请求），所以独立于 Web 服务器和浏览器。Ajax 的关键就是 JavaScript 脚本本身和 XMLHttpRequest 对象。

熟练掌握 Ajax 的使用，需要理解其工作原理和基本原则。Ajax 的工作原理相当于在用户和服务器之间增加一个中间层，使用户操作与服务器响应异步化。这样就可以将服务器负担的一些工作转接到客户端，利用客户端闲置的处理能力来处理，从而减轻服务器和带宽的负担，达到节约 ISP（Internet Service Provider，因特网服务提供商）的空间及带宽租用成本的目的。图 9-1 演示了 Ajax 的这种工作原理。

通过图 9-1 的 Ajax 的运行机制可见，使用 Ajax 可以在很大程度上减轻服务器的负担，减少服务器对用户的响应时间。

Ajax 的原则是“按需取数据”，可以最大程度地减少冗余请求和响应对服务器造成的负担。Ajax 的目标是通过 Web 浏览器交付具有良好可用性的独占应用，以满足提高用户的生产力和

通过网络来共享数据两方面的需要，同时还具备 Web 应用集中维护的优点。为了成功实现这个目标，需要以一种完全不同的方式来思考 Web 页面和应用，具体如下所示。

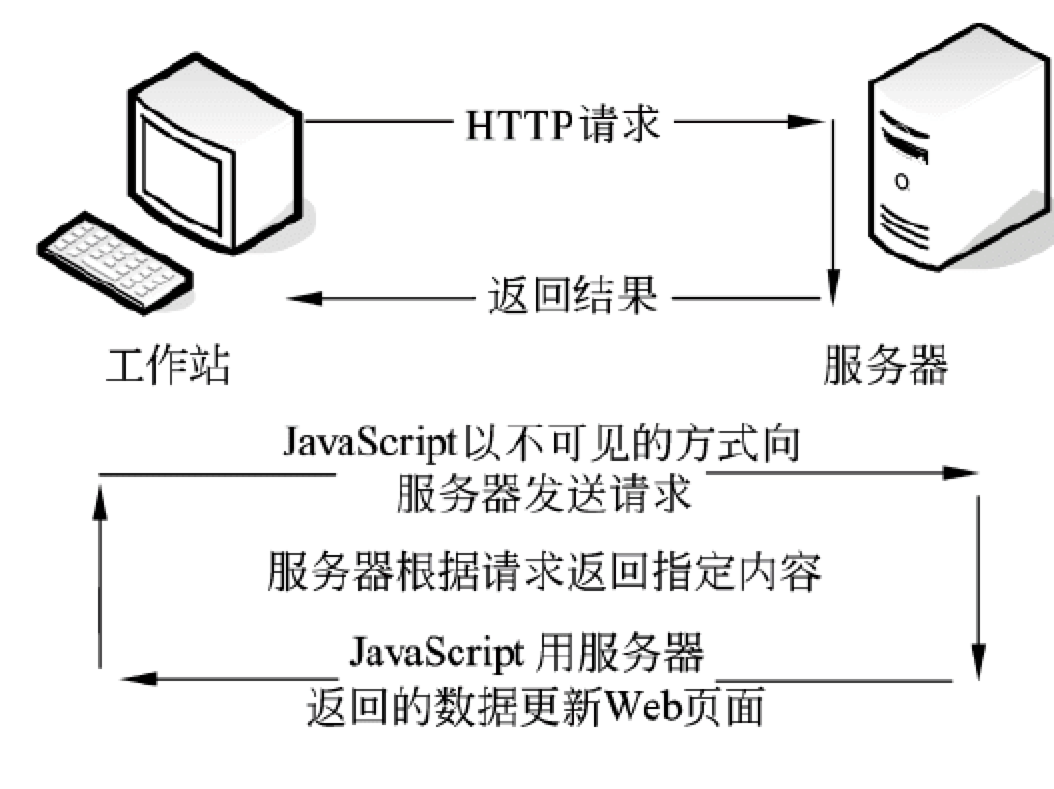


图 9-1　Ajax 的工作原理

- 浏览器中的是应用，而不是内容。
- 服务器交付的是数据，而不是内容。
- 用户和应用的交互是连续的，大部分对于服务器的请求是隐式的而不是显式的。
- 代码库是巨大的、复杂的、而且组织良好，这个特点对于架构来说非常重要。

9.1.4　Ajax 核心内容

Ajax 核心是 XMLHttpRequest 对象，该对象在 Internet Explorer 5 中被首次引入，是一种支持异步请求的技术。通过 XMLHttpRequest 对象，用户可以使用 JavaScript 向服务器提出请求并处理响应，并且不阻塞用户。

Ajax 在创建 Web 站点时，在客户端执行屏幕更新，为用户提供很大的灵活性。下面是使用 Ajax 可以完成的功能。

- **提升站点的性能**

通过减少从服务器下载的数据量实现。例如在某购物车页面，当更新篮子中的一项物品的数量时，会重新载入整个页面，这就必须下载整个页面的数据。如果使用 Ajax 计算新的总量，服务器只会返回新的总量值，因此所需的带宽仅为原来的百分之一。

- **消除每次用户输入时的页面刷新**

例如在 Ajax 中，当用户在分页列表上单击【下一页】按钮时，服务器数据只刷新列表而不是整个页面。使用 Ajax 技术中的 XMLHttpRequest 对象发送请求，服务器可以返回一个只包含需要返回的特定信息长度的字符，而不是发送整个页面的数据来刷新页面。即便用户使用的是高速 Internet，这样的差别有时也会非常大。

- **实现对表格的直接编辑**

直接编辑表格数据，而不是要求用户导航到新的页面来编辑数据。对于 Ajax 当用户单击【编辑】按钮时，可以将静态表格刷新为内容可编辑的表格。用户单击【提交】按钮之后，就可以发出一个 Ajax 请求来更新服务器，并刷新表格使其包含静态、只读的数据。

9.2 Ajax 结构及其意义

Ajax 允许更简洁的 Web 应用程序，并为生成新的应用程序奠定基础。Ajax 使 Web 应用程序的交互性变得更强，响应更及时、更快速且更友好。交互和响应（操作缓慢时的反馈）都更妥善地得到处理，便于用户对页面进行更多操作。Ajax 允许用户和页面间的交互与浏览器和服务器间的通讯分离开。Ajax 的体系结构如图 9-2 所示。

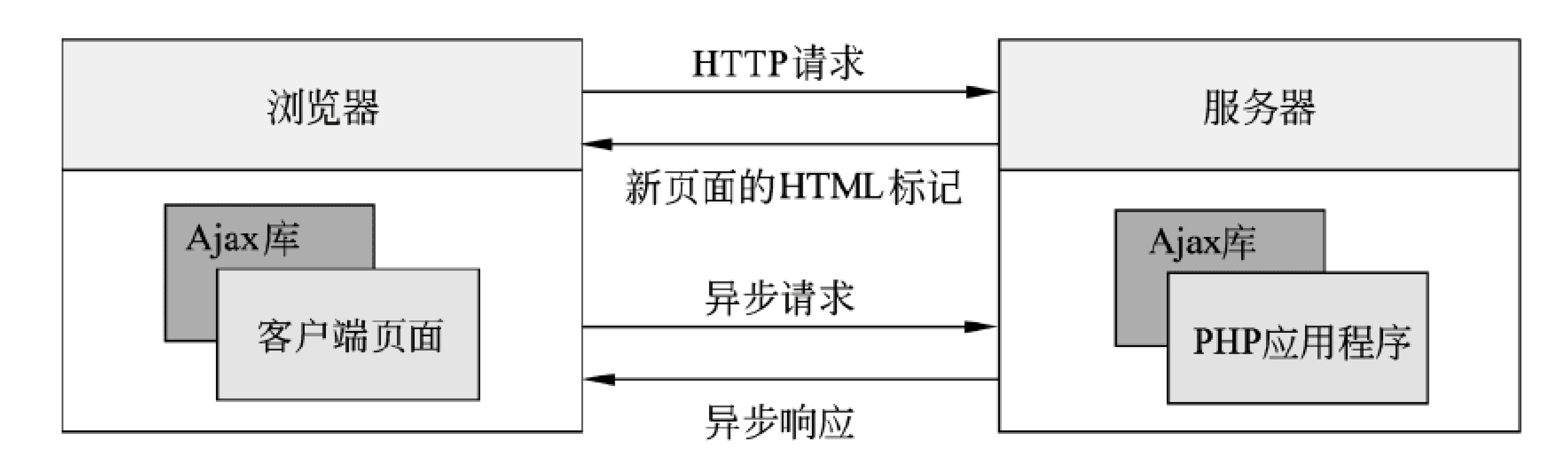


图 9-2　Ajax 的体系结构

Ajax 应用程序使用负责对服务器发出调用的客户端框架，Ajax 服务器端框架则负责请求数据处理并将其返回客户端。这通常是 JavaScript Object Notation（JSON）数据流，但也可使用其他格式，如 XML、RSS 和 CSV。

客户端接收返回数据，并使用 JavaScript 更新界面。服务器通过返回以指定格式编码的原始数据，对请求做出响应。带宽消耗会降至最低，应用程序速度会提高（因为完成请求所用时间较少），而且界面更新也能在无可见回发的情况下生效。但是在解决众多问题的同时，客户端操作的增加也带来新的问题，例如新的编码实践、安全隐患以及可访问性问题等。

9.2.1　传统 Web 应用解决方案

传统的 Web 应用采用同步交互过程，这种情况下，用户首先向 HTTP 服务器发出一个行为或请求。然后服务器执行某些任务，再向发出请求的用户返回一个 HTML 页面。这是一种不连贯的用户体验，服务器在处理请求时，用户多数时间处于等待的状态，屏幕内容也是一片空白。

在传统 Web 应用解决方案中，如果用户使用浏览器浏览网页，当页面刷新很慢时，用户的浏览器在干什么呢？屏幕显示什么内容呢？通常情况下，用户的浏览器在等待刷新，屏幕内容是一片空白，而用户只能在屏幕前苦苦等待浏览器的响应。开发人员为了克服这种尴尬局面，不得不在每一个可能需要长时间等待响应的页面上增加一个 DIV 标记，在需要时显示类似“系统正在处理您的请求……”等信息以提示用户，从而改善用户操作体验。图 9-3 和图 9-4 所示为传统 Web 应用程序的结构和模型。

自从采用超文本作为 Web 传输和呈现之后，就一直使用这样的传输方式。当负载比较小时，这种方式并没有表现出什么问题。可是当负载比较大时，响应时间可能要很长，1 分钟、2 分钟或者数分钟的时间，如果由于响应时间过长，而超过最大响应时间，那么服务器将返回

页面不可用的信息。另外，有些时候只是想改变页面中某一部分的数据，而并非要重新加载整个页面。当软件设计越来越讲究人性化时，这么糟糕的用户体验简直与这种原则背道而驰。为什么总是要让用户等待服务器返回数据呢？至少应该通过一定的方法来减少用户等待时间。现在，除了程序设计、编码优化和服务器调优之外，还可以采用 Ajax。

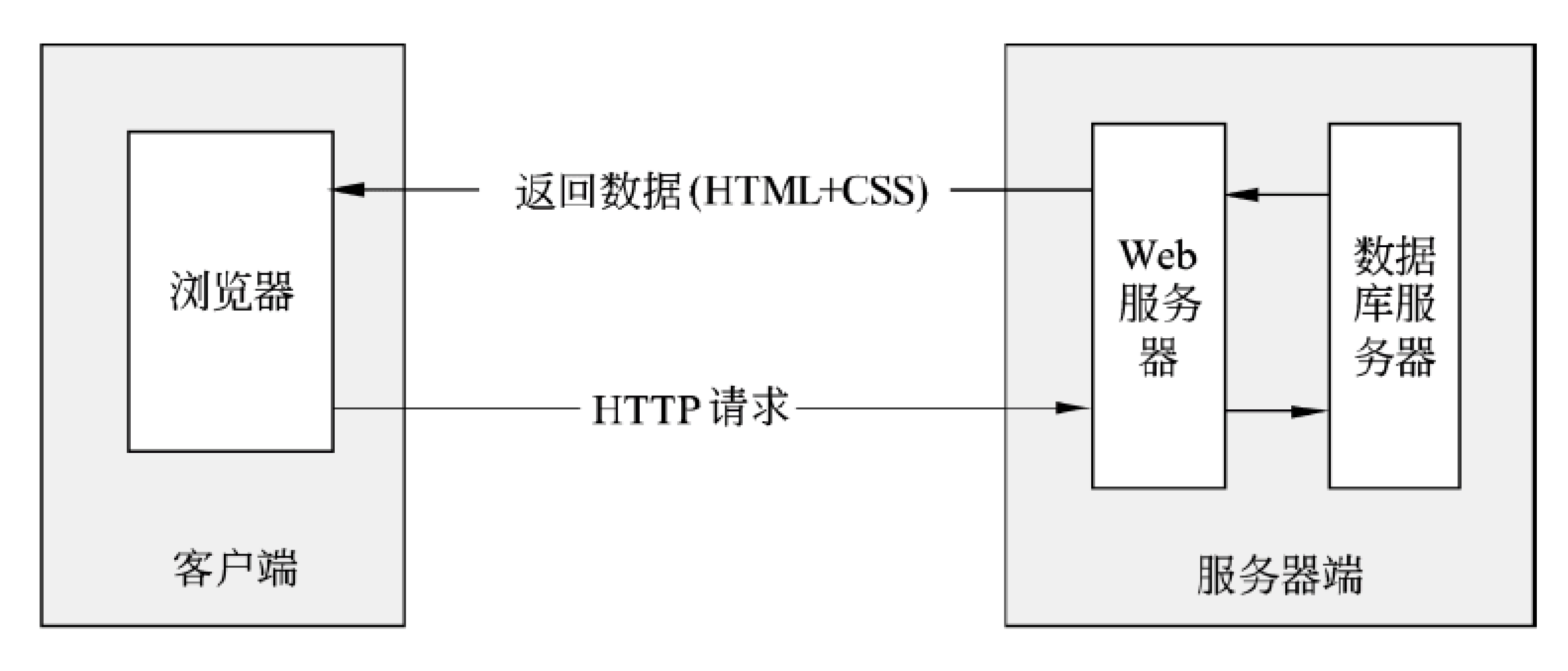


图 9-3　传统 Web 应用程序的结构

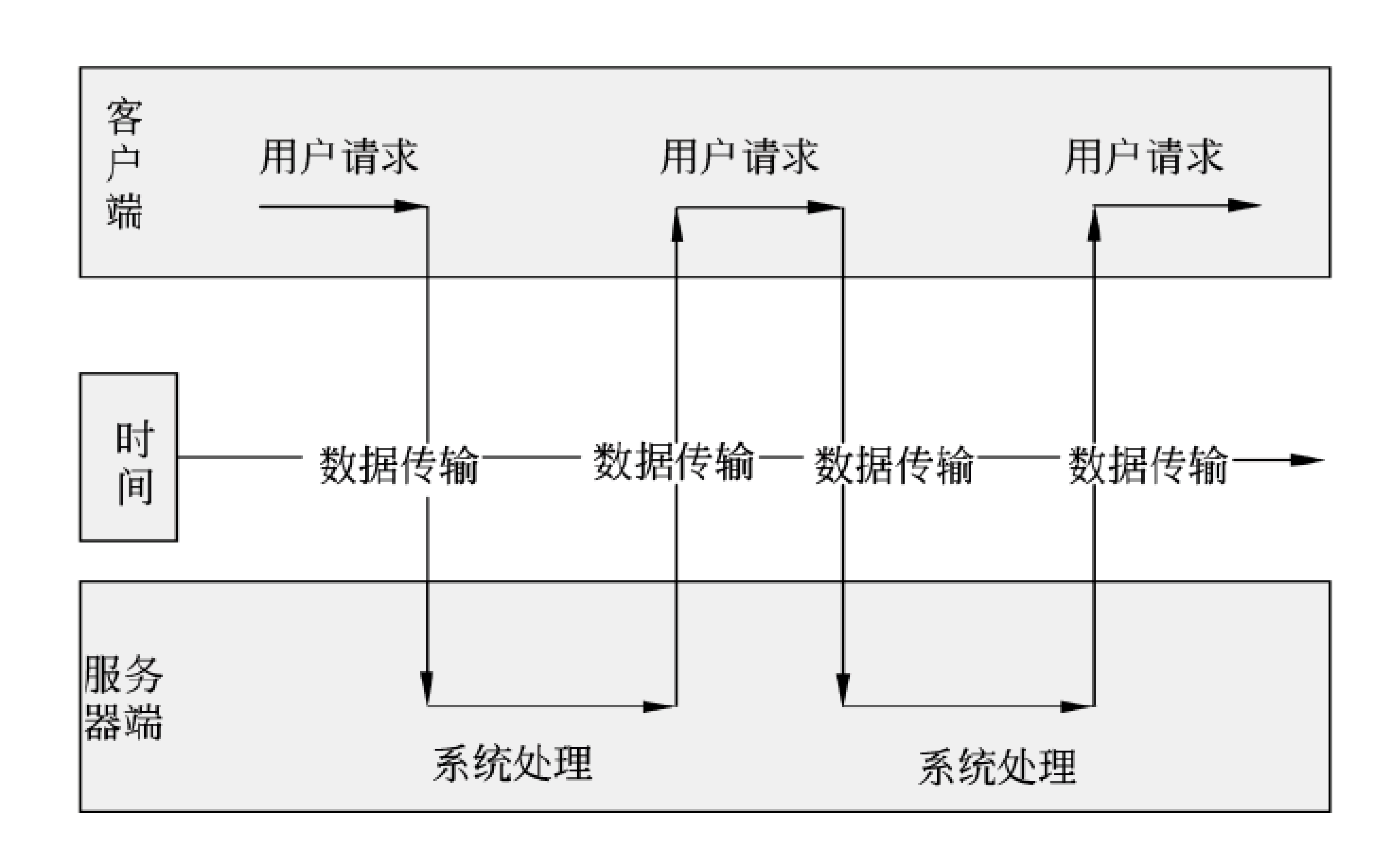


图 9-4　传统 Web 应用程序的模型

Ajax 应用可以仅向服务器发送并返回必需的数据，同时使用 SOAP 或其他一些基于 XML 的 Web service 接口，并在客户端采用 JavaScript 处理来自服务器的响应。因为在服务器和浏览器之间交换的数据大量减少，结果就能看到响应更快的应用。同时很多的处理工作可以在发出请求的客户端机器上完成，所以 Web 服务器的处理时间也减少。

9.2.2　Ajax 解决方案的优势

使用 Ajax 可以实现无刷新更新页面，减少用户心理和实际的等待时间，当要读取大量数据时，不会像重新加载那样出现白屏的情况。Ajax 使用 XMLHttpRequest 对象发送请求并得到服务器响应，在不重新载入整个页面的情况下，使用 JavaScript 操作 DOM 更新页面。所以在读取数据的过程中，用户所面对的不是白屏，而是原来的页面内容（也可以加一个 Loading 的提示框让用户知道处于读取数据过程），只有当数据接收完毕之后才更新相应部分的内容。

这种更新是瞬间的，用户几乎感觉不到。Ajax 的优势如下所示。

- 带来更好的用户体验。
- 可以将以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。
- 可以调用外部数据。
- 基于标准化并被广泛支持的技术，不需要下载插件或者小程序。
- 进一步促进页面呈现和数据分离。

可以看出 Ajax 技术的优势相当明显，但何时应该使用 Ajax 技术呢？这是一个很重要的问题。如果 Ajax 使用过渡，那么就会有数千行 JavaScript 代码在客户端的浏览器上运行，可能会让用户感觉速度很慢，如果这些脚本编写不当，就会很快失去控制，特别是当通信量很大时。所以 Ajax 技术的使用也有一定的范围，例如用于改善用户操作体验等。

Ajax 与传统 Web 程序的区别就在于：传统 Web 程序发送带参数的 HTTP 请求到服务器，服务器根据参数重新生成 HTML 页面返回客户端。而 Ajax 通过使用 XMLHttpRequest 隐藏地向服务器端发送 HTTP 请求和参数，通过 JavaScript 获取 HTTP 请求返回的数据，然后通过 JavaScript 动态地给客户端 HTML 文档添加节点和文本。

9.2.3 Ajax 的应用

Ajax 的主要特点在于异步交互更新 Web 页面的局部信息。因此比较适用于交互较多、读取数据频繁、数据传输量比较小的 Web 应用。如表 9-1 所示的几种情况。

表 9-1 Ajax 的应用情况

应用情况	说明
基于表单的简单用户交互	例如用户注册验证、数据格式验证等，如果采取整页回发到服务器的做法，不仅传输数据量大、服务器负担重、响应时间长，而且用户体验很差。在验证结束后，由于某些数据错误返回注册页面时，先前输入的数据都已不存在，必须重新填写。而采用 Ajax 技术后，这些操作在用户填写其他信息时，已由 XMLHttpRequest 对象异步完成，极大地改善了用户操作体验
实时更新的页面信息	如聊天室信息、在线统计、股票的涨跌等都需要实时地反映数据的变化。采用 Ajax 技术定时异步访问服务器，可以获得最新信息显示给用户，且可以避免整个页面的刷新
菜单导航	如多级联动菜单、树状导航等都可以采用 Ajax 技术来实现按需读取数据，这样可以避免每次变动都需要回发到服务器，从而节省带宽资源，提高响应速度，又减少显示所有数据时所要消耗的带宽资源
评论、选择、投票	这几种情况传输的数据量非常小，将整个页面回发到服务器是没有必要的。如果采用 Ajax 技术，用户在执行完相关操作后，将异步与服务器进行自动交互，而用户同时可以继续执行其他操作

9.2.4 Ajax 相关技术简介

从 Ajax 的全称上可以看出 Ajax 其实并不是一种技术，而是几种技术的组合体。每种技

术都有其独特之处，组合在一起就成为一个功能强大的新技术。Ajax 的内容组成如图 9-5 所示。

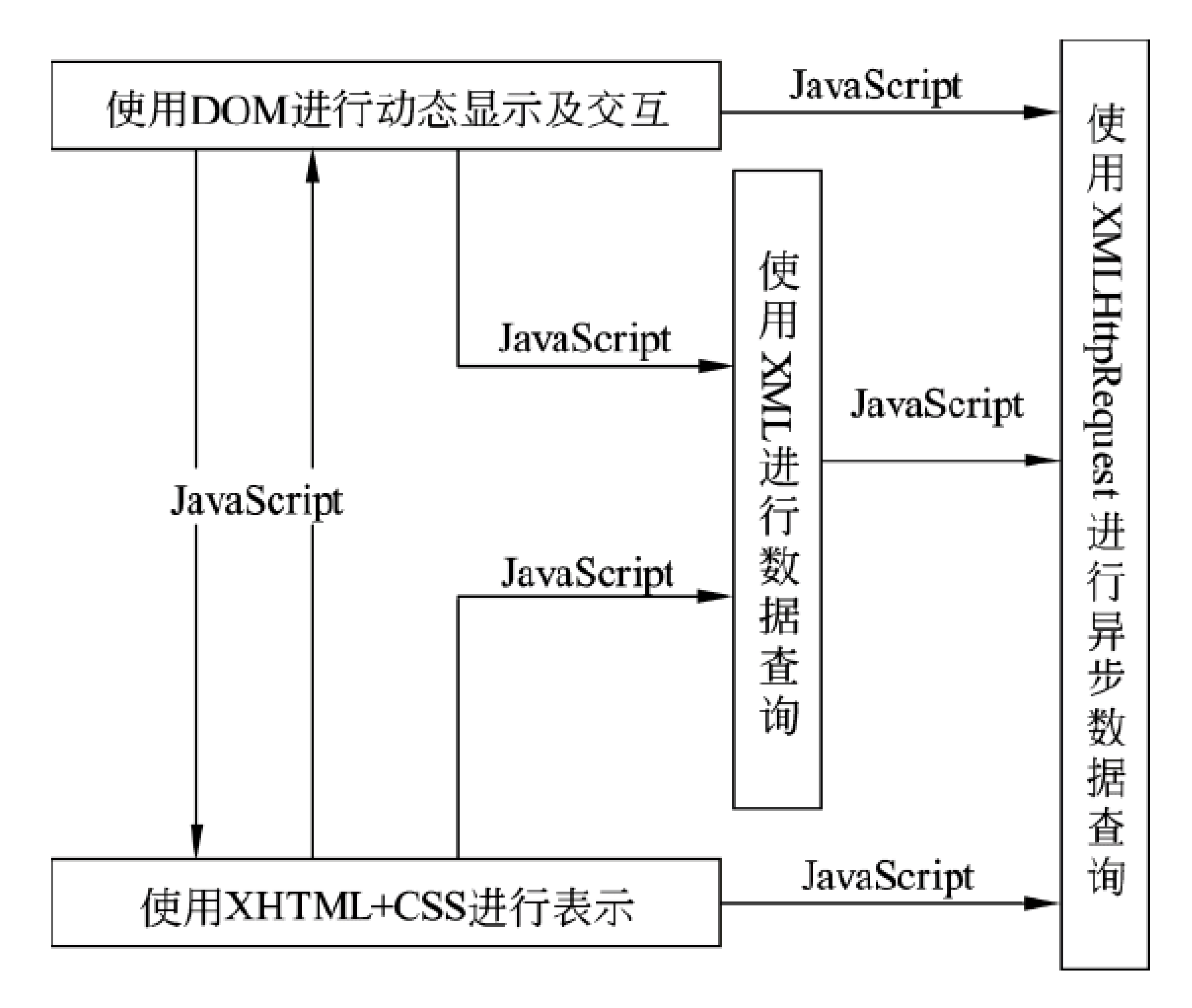


图 9-5　Ajax 相关技术

从图 9-5 中可以看出，Ajax 由以下内容组成。

- 基于 Web 标准的 XHTML 可扩展标识语言和 CSS 进行表示。
- 使用 DOM 进行动态显示及交互。
- 使用 XML 和 XSLT 进行数据交换及相关操作。
- 使用 XMLHttpRequest 对象进行异步数据查询、检索。
- 使用 JavaScript 脚本语言将所有的东西绑定在一起。

Ajax 应用以支持上述技术的 Web 浏览器作为运行平台。这些浏览器目前包括：Mozilla、Firefox、Internet Explorer、Opera、Konqueror 以及 Safari。但是 Opera 对 Ajax 的支持并不完美，因为该浏览器不支持 XSL 格式对象，也不支持 XSLT。

要使 Ajax 成功运行，需要使用客户端与服务器端两种技术互相配合。客户端采用 JavaScript 向服务器端发送请求，服务器端根据请求返回指定内容。客户端的 XMLHttpRequest 对象以异步方式向服务器发送请求，当服务器端执行请求时，用户可以继续前台的其他工作。

如果只有客户端而没有与之对应的服务端响应用户请求也不行，所以也需要服务端对客户端的 JavaScript 请求做出回应。如果服务器返回 XML，则客户端的 JavaScript 可以非常方便地读取其中的内容。服务器管理者也可以根据需要返回特定的数据类型，甚至可以只是一个文本。

9.3　第一个 Ajax 实例

Ajax 处理过程中的第一步是创建一个 XMLHttpRequest 对象，使用 HTTP 方法（GET 或 POST）来发送请求，并将目标 URL 设置到 XMLHttpRequest 对象上。

当发送 HTTP 请求之后，并不希望浏览器挂起以等待服务器的响应，而是希望通过页面继续响应用户的操作，并在服务器响应真正到达后处理。要完成上述步骤，首先需要向 XMLHttpRequest 对象注册一个回调函数，并异步发送 HTTP 请求，当服务器响应到达时，回调函数将被调用。

下面通过第一个 Ajax 实例的制作，使读者了解 Ajax 应用程序的开发步骤，加深对 Ajax 运行机制的理解。实例比较简单，主要是实现了在不干扰用户输入的情况下，验证用户名是否唯一。

首先创建一个 HTML 页面，该页面用于填写用户的信息。当输入用户名之后，使用 Ajax 及时地验证用户名是否可用。在不进行页面刷新的情况下给出相应的提示信息。下面给出 HTML 页面局部代码，如代码 9.1 所示。

代码 9.1　Login.html

```
<form action="#">
<center>
<h3>Ajax+php 验证用户名是否可用</h3>
<table>
<tr>
<td>用户名： <input type="text" name="UserID" id="UserID" onchange=
"startRequest()" />
<span id="span"></span></td>
</tr>
<tr>
<td>密  码：  <input type="text" name="Pwd" id="Pwd"/></td>
</tr>
<tr>
<td>
         
<input type="button" name="Reg" id="Reg" value="注册"/>
<input type="button" name="Log" id="Log" value="登录"/>
</td>
</tr>
</table>
</center>
<form>
```

Ajax 的执行原理实际上就是调用 XMLHttpRequest 对象，那么首先必须创建这个对象，对象创建好以后，就可以发送 XMLHttpRequest 请求。发送请求可以使用 GET 方法和 POST 方法。GET 方法适用于页面中的任何元素，包括表单等。在实际应用中，对表单的操作比较多，针对表单，更多使用的是 POST 方法。下面实现创建 XMLHttpRequest 对象、发送请求和处理请求的功能，如代码 9.2 所示。

代码 9.2　初始化 Ajax

```
<script type="text/javascript">
```

```
var xmlHttp;
function createXMLHttpRequest()
{
  //在 IE 下创建 XMLHttpRequest 对象
  try
  {
    xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
  }
  catch(e)
  {
    try
    {
      xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
  catch(oc)
  {
    xmlHttp = null;
  }
 }
  //在 Mozilla 和 Safari 等非 IE 浏览器下创建 XMLHttpRequest 对象
  if(!xmlHttp && typeof XMLHttpRequest != "undefined")
 {
  xmlHttp = new XMLHttpRequest();
 }
return xmlHttp;
}
//发送请求
function startRequest()
{
  //获取用户输入的信息
  var UserID = document.getElementById("UserID").value;
  //输入的用户名是否为空
  if(UserID!="")
    {
     //创建 XMLHttpRequest 对象
     createXMLHttpRequest();
     var url = "Login.php?UserID="+encodeURI(UserID); //指定 url
     xmlHttp.open("GET",url,true);
     xmlHttp.onreadystatechange = handleStateChange;  //指定回调函数
     xmlHttp.send(null);
    }
   else
   {
     alert("请填写您的用户名!");
   }
```

```
}
//回调函数
function handleStateChange()
{
   //判断 readyState 的状态
   if(xmlHttp.readyState<4)
   {
   span.innerHTML="正在读取数据...";
   }
   //请求完成 (readyState 值为 4)
   if(xmlHttp.readyState==4)
   {
   //响应完成 (HTTP 状态为 200)
    if(xmlHttp.status == 200)
     {
         span.innerHTML=xmlHttp.responseText;
     }
      else{
        alert("错误，请求页面异常!");
         }
    }
}
</script>
```

上述代码中，首先声明一个全局变量 xmlHttp，然后调用 createXMLHttpRequest()函数，该函数功能就是创建 XMLHttpRequest 对象，并根据不同的浏览器对其进行相应的初始化。接着判断 xmlHttp 对象是否创建并初始化成功。最后返回创建好的 xmlHttp 对象。

startRequest()函数用于发送 XMLHttpRequest 请求。首先获取用户输入的用户名，然后判断用户名是否为空，如果不为空，调用 createXMLHttpRequest()函数创建 XMLHttpRequest 对象，并指定回调函数。否则弹出一个对话框，提示“请填写您的用户名!”。

handleStateChange()函数首先判断 readyState 的状态，如果状态小于 4，则在页面中输出“正在读取数据...”，当请求完成加载（readyState 值为 4）并且响应已经完成（HTTP 状态为 200）时，在 span 标签中显示信息。否则弹出一个对话框，提示信息“错误，请求页面异常!”。

HTML 页面设计好以后，接下来设计 PHP 页面。首先需要设置 PHP 的字符集为 GBK，否则页面运行之后显示的是乱码。然后定义一个变量并使用 GET 方法获取输入的用户名，接着判断输入的用户名是否等于“开心果冻”，如果等于，则在页面中输出“已被注册”，否则输出“还没有被注册!”。PHP 页面的具体实现如代码 9.3 所示。

代码 9.3　Login.php

```
<?php
//设置 PHP 的字符集
header('Content-Type:text/html;charset=GBK');
//令程序先等待一秒
```

```
sleep(1);
//获取输入的用户名
$tmp=$_GET["UserID"];
//如果在客户端输入“开心果冻”，那么就显示已被注册
if ($tmp=='开心果冻')
{
  echo("已被注册");
}else{
  echo("还没有被注册！");
}
?>
```

将该段代码保存为 Login.php。此时，程序已经编写完成。接下来在浏览器中查看页面运行效果。在浏览器地址栏中输入“http://localhost:8080/Login.html”则可以访问到 Login.html 页面。在用户名文本框中输入信息，则验证该用户名是否可用。页面运行结果如图 9-6 和图 9-7 所示。

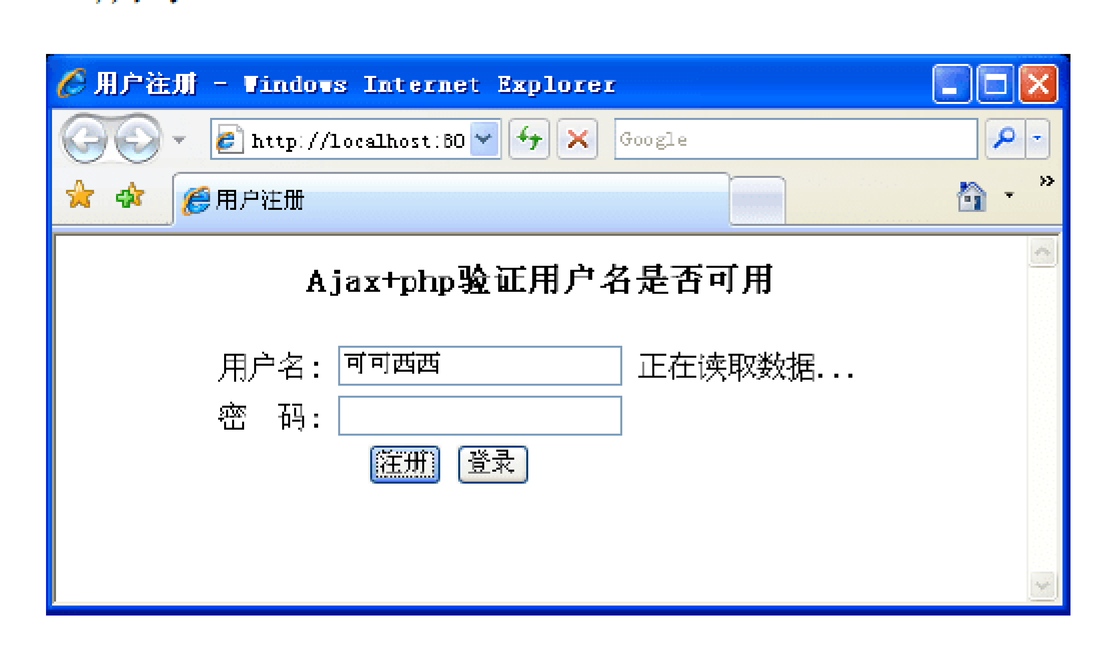


图 9-6　等待验证



图 9-7　验证结果

第10章 CSS



一个优秀的网站不仅需要具有一套完整的内容结构，巧妙且美观的外观布局及合理的内容搭配，还需要一个统一的风格。这样才能增强网站的吸引力，提高访问量，从而在众多的网站中脱颖而出。CSS 就可以实现这种功能，使用 CSS 可以有效地对页面的布局、字体、颜色、背景和其他效果实现精确的控制。只需做一些简单的修改，就可以改变同一页面的不同部分，或者统一网站页面的外观和格式。

本章将详细讲解 CSS，包括 CSS 基本概念、使用选择符定义 CSS 规则、在 HTML 中应用 CSS、CSS 修饰页面基本属性和区块属性，还介绍了如何使用位置属性对页面进行布局以及一些其他 CSS 属性。



- 了解 CSS 概念
- 掌握选择符的定义
- 熟悉应用 CSS 的 4 种方式
- 掌握 CSS 基本属性在页面中的作用
- 掌握文本、背景和列表的使用
- 掌握 CSS 对区块的设置及作用
- 掌握如何定位和制作布局
- 熟悉 CSS 的单位和鼠标指针
- 了解 CSS 滤镜

10.1 CSS 概述

CSS 功能非常强大，不仅弥补了 HTML 的不足，还可以将网页变得更加美观，维护更加方便。与 HTML 一样，CSS 也是一种标记语言，而且很多属性都来源于 HTML，可运行于所有浏览器中。通过本节，读者将了解更多关于 CSS 的知识，像如何创建一个 CSS 规则，如何在页面中应用 CSS 等。

10.1.1 CSS 简介

CSS（Cascading Style Sheets）中文译为层叠样式表，是用于增强或者控制网页样式并允

许将样式信息与网页内容分离的一种技术。目前，几乎所有的浏览器都支持 CSS，而且已成为网页设计中必不可少的技术之一。

CSS 和 HTML 一样，都是由 W3C（World Wide Web Consortium）组织定义、发布和维护的。目前被广泛地使用的是 1998 年 W3C 发布的 CSS 2.0 版本。W3C 对 CSS 的定义是：CSS 是一种标记语言（与 HTML 相同），不需要编译，可以由浏览器直接解释执行，以扩展名为“.css”的文本文件存在。

CSS 最大优势是仅仅通过一个 CSS 样式表就能够使网页设计者控制所有出现在 Web 中的外观及布局，并且可以为每个标记的元素和应用该元素的每个页面定义一个需要的样式，从而实现全面的改变。简单地改变样式，所有与之相关的元素都会自动更新。

采用 DIV+CSS 进行网页重构相对于传统的 TABLE 网页布局而言，具有以下 3 个显著特点。

- **表现和内容相分离**

将设计分离出来放在一个独立样式文件中，HTML 文件中只存放内容信息。这样的页面对搜索引擎更加友好，实现内容、结构和表现的分离。

- **提高页面浏览速度**

对于同一个页面视觉效果，采用 DIV+CSS 重构的页面容量要比 TABLE 的页面文件容量小得多，前者一般只有后者的 1/2 大小。因此，浏览器就不用去编译大量冗长的标签和理解复杂的页面结构。

- **易于维护和改版**

只需简单地修改 CSS 文件就可以重新设计整个网站的页面。

在给页面或页面元素应用样式时，CSS 提供了 3 种方式，即 CSS 的 3 种类型：类、标签和高级。

- **类**

为了对网页样式定义得更为精确，有必要让相同的选择符（一般情况下为页面标记）也能分类，并能按照不同的类别来进行不同的样式设计。类就是这样一种新的样式表示符，适用于修饰页面中特定的区域。

- **标签**

标签是将页面文件中的 XHTML 标记重定义。用标签定义的 CSS 样式设置完成后，该 CSS 样式马上生效。例如定义段落标记<p>样式，则可以使用标签类型。

- **高级**

高级是指将 CSS 样式用于特定的 XHTML 标记组合或 ID 名，例如伪类选择符和 ID 选择符。关于伪类选择符，最常用的标记就是锚标记<a>。还可以与 XHTML 标记进行组合，例如 td img，这个 CSS 样式将用于网页上所有 td 标记内的 img 标记。将 CSS 样式应用于 ID 名，则可以使用 ID 选择符，例如可使用#Content，则这个 CSS 样式将应用于网页上所有带 ID="Content"的 XHTML 标记。

10.1.2　定义 CSS 规则

所有 CSS 样式表的基础就是 CSS 规则。每个 CSS 规则是一条单独的语句，用于定义如何

设计样式，以及应该如何应用这些样式。所以，样式表由规则列表组成，浏览器由它来确定页面的显示效果。

1. 基本规则

CSS 规则由 3 个部分构成：选择符（selector）、属性（property）和值（value）。语法如下：

```
选择符 { 属性 : 值 }
```

选择符可以采用多种形式，但一般为文档中的 HTML 标记，例如 body、table 和 p 等。属性则是选择符指定的标记所包含的属性。值是指定属性的值。如果定义选择符的多个属性，则属性和属性值为一组，组与组之间用分号（;）隔开。

下面就定义了一条样式规则，该样式规则是为 p 标记提供样式，color 为指定文字颜色的属性，red 为属性值，表示含义为 p 标记中的文字使用红色。

```
p {color:red}
```

如果属性值由多个字符串及空格组成，那么该属性值就必须使用双引号（""）。同时，一条样式规则中的多个属性使用分号（;）进行分隔。例如，以下样式都正确：

```
p {font-family:" Times New Roman 宋体"}
p {font-family:"宋体"; color:red; font-size:40px; float:left; }
```

对于使用相同样式（例如背景颜色、字体大小）的标记，可以将选择符组合起来形成选择符组，各选择符之间用逗号（,）分隔。例如下面定义的样式同时应用到 p、div 和 ul 标记：

```
p,div,ul{ argin:0 auto;padding:0;font-size:12px;list-style: none;}
```

为了便于阅读和维护，建议读者在编写样式时使用分行的格式。例如将前面样式应用分行格式后的形式如下：

```
p,div,ul{
   argin:0 auto;
   padding:0;
   font-size:12px;
   list-style: none;
}
```

2. 类选择符定义样式

除了可以为多个标记指定相同样式外，还可以使用类选择符来定义一个样式，这种方法同样可以应用到不同的标记上。定义类选择符方法是在自定义样式的名称前面加一个句点(.)。如下代码使用类选择符定义了一个名为“.Header”的样式：

```
.Header {
     color:#006;
     font-size:24px;
     text-align:center;
```

```
    font-weight:bold;
}
```

样式定义了字体颜色、大小、对齐方式以及加粗。要使用类选择符定义的样式，只需将标记的 class 属性指定为样式名称。例如要在 p、span 和 div 标记里使用前面的.Header 样式，可以使用如下代码：

```
<p class="Header">打造一流 IT 学习乐园　推进无纸化教学进程</p>
<span class="Header">打造一流 IT 学习乐园　推进无纸化教学进程</span>
<div class="Header">打造一流 IT 学习乐园　推进无纸化教学进程</div>
```

使用这种方法，可以很方便地在任意元素上套用预先定义好的类样式，从而实现相同的样式外观。

3. ID 选择符定义样式

在页面中，元素的 ID 属性指定了某个唯一元素的标识，同样 ID 选择符可以用来对某个特定元素定义独特的样式。ID 选择符的应用和类选择符类似，只要将 class 换成 ID 即可。例如在 span 标记中通过使用 ID 属性来引用样式“Header”：

```
<span id="Header">打造一流 IT 学习乐园　推进无纸化教学进程</span>
```

与类选择符不同，使用 ID 选择符定义样式时，需在 ID 名称前加上一个“#”号。例如，对于上述语句，使用 ID 选择符定义样式代码如下所示：

```
# Header {
    color:#006;
    font-size:24px;
    text-align:center;
    font-weight:bold;
}
```

4. 伪类选择符

伪类选择符可以看作是一种特殊的类选择符，是能被支持 CSS 的浏览器自动识别的特殊选择符。伪类选择符的最大作用就是可以对链接的不同状态定义不同效果。关于伪类选择符，最常用的标记就是 a 标记，即超链接的伪类选择符。表示超链接 4 种不同的状态：未访问的链接（link）、已访问的链接（visited）、激活链接（active）和鼠标停留在链接上（hover）。代码如下所示：

```
a:link{color:#FF0000; text-decoration:none}
a:visited{color:#00FF00; text-decoration:none}
a:hover{color:#0000FF; text-decoration:underline}
a:active{color:#FF00FF; text-decoration:underline}
```

上面的样式表示该链接未访问时颜色为红色且无下划线，访问后是绿色且无下划线，激活链接时为蓝色且有下划线，鼠标放在链接上为紫色且有下划线。

5．混合方式

严格地讲，这不算是样式定义的新方式。在 CSS 中任意一种定义方式都可以进行组合。类选择符可以和 ID 选择符组合使用，伪类选择符也可以和类选择符组合使用，在同一页面中完成几组不同的链接效果。如下的一条样式定义了 5 个名称：

```
H2,td.news,p#item,.MainText,li a{
    font-size:12px;
    text-align:center;
    font-weight:normal;
    border-top:#ccc 1px dashed;
    color:#666;
    height:24px;
    margin:3px 5px 8px 0;
    background:#f5f5f5;
}
```

要应用上述样式有很多方法：可以使用 H2 标记选择符；类选择符应用.MainText，标准选择符应用 p，ID 选择符应用“p#item”；或者在 td 标记中使用 class 应用“td.news”等。这几种应用方法，由于定义的 CSS 样式规则相同，因此运行结果也相同，代码如下所示：

```
<h2>打造一流 IT 学习乐园 推进无纸化教学进程</h2>
<table><tr><td class="news">
打造一流 IT 学习乐园 推进无纸化教学进程
</td></tr></table>
<p id="item">打造一流 IT 学习乐园 推进无纸化教学进程</p>
<span class="MainText">打造一流 IT 学习乐园 推进无纸化教学进程</span>
<ul>
<li><a href="#">窗内网</a></li>
<li>打造一流 IT 学习乐园 推进无纸化教学进程</li>
</ul>
```

10.1.3 应用 CSS

由于 CSS 样式表以普通文本形式存在，因此网页设计者可以使用任何一种文本编辑工具来进行编辑，例如记事本和写字板。当然也可以使用一些专业的 HTML 编辑工具，例如 Dreamweaver 和 FrontPage，来编辑 CSS 文档。

在 CSS 中可以使用 4 种不同的方法将样式表的功能应用到网页中。包括：直接定义元素的 style 属性、定义内部样式表、嵌入外部样式表和连接外部样式表，这 4 种方法又可以归纳为使用内部样式表和外部样式表两大类。

1．连接外部样式表

连接外部样式表是 CSS 样式表应用中最好的一种形式。将 CSS 样式表代码单独编写在一

个独立文件中，由网页进行调用，多个网页可以调用同一个外部样式表文件。因此，能够实现代码的最大化重用以及网站文件的最优化配置。

具体是指在外部定义 CSS 样式表并形成以.css 为扩展名的文件，然后在页面中通过<link>链接标记链接到页面中，而且该链接语句必须放在页面的 head 标记区，如下所示：

```
<link rel="stylesheet" type="text/css" href="style.css" />
```

link 标记的 rel 属性指定链接到样式表，type 表示样式表类型为 CSS 样式表，href 指定了 CSS 样式表的所在位置，这里使用的是相对路径。如果 HTML 文档与 CSS 样式表没有在同一路径下，则需要指定样式表的绝对路径或者引用位置。

例如存在一个外部样式表文件 Style1.css，包含了代码 10.1 所示的内容。

代码 10.1　外部样式表文件：Style1.css

```
body {
    margin:0;
    padding:0;
    font-size:14px;
}
.content {
    margin-left:10px;
    line-height:24px;
}
#wrap {
    margin: 20px auto 10px auto;
    width: 600px;
    background: #fff;
    padding: 10px;
    border: 5px solid #000;
    text-align: left;
}
h1 {
    color:#006;
    font-size:24px;
    font-weight:bold;
    text-align:center;
    margin-bottom:0px;
}
h2 {
    font-size:12px;
    text-align:center;
    font-weight:normal;
    border-top:#ccc 1px dashed;
    color:#666;
    height:24px;
```

```
        margin:3px 5px 8px 0;
        background:#f5f5f5;
    }
    p {
        text-indent: 20px;
    }
```

接下来，在同一目录中创建一个 HTML 文件，再使用 link 标记将上述 Style1.css 连接到文件中。最后，将 HTML 文件保存为“外部样式 1.html”，并添加代码 10.2 所示内容。

代码 10.2　在 HTML 中连接外部样式表：外部样式 1.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>窗内网</title>
<link href="Style1.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="wrap">
    <h1>窗内网简介</h1>
    <h2>来源: 窗内网 | 浏览次数: 56 次 | 发布时间: 2009-01-18</h2>
        <p>窗内网是汇智科技旗下的在线教育平台，为入门级学员提供专家训练营系列课程，帮助学
        员系统地掌握流行技术；为提高级学员提供专题课程、精品讲座和企业学院系列课程，帮助快
        速学习新技术。</p>
</div>
</body>
</html>
```

直接打开“外部样式 1.html”文件在浏览器中查看效果，如图 10-1 所示。



图 10-1　应用外部样式表效果

2. 内部样式表

内部样式表是指将 CSS 样式表直接在 HTML 页面代码的 head 标记区定义，由“<style

type="text/css" >”标记开始至“</style>”结束。示例代码如下所示：

```
<style type="text/css" >
   body{ margin: 0 auto;  padding: 0;}
   form, ul, ol, li, dl, dt, dd{ font-family:Arial,"宋体";font-size:12px;
   text-align:center;}
</style>
```

例如在“外部样式 1.html”文件的基础上，在 head 标记区内定义一个带图像的内部 CSS 样式，如代码 10.3 所示。

代码 10.3　内部样式

```
<style type="text/css">
#logo {
    background-image: url(chuangnei.gif);
    background-repeat: no-repeat;
    height: 58px;
}
</style>
```

然后，在页面<body>标记区的<h1>标记前添加“<div id="logo"></div>”语句，应用上述内部样式。最后，将页面另存为“内部样式 1.html”并执行，效果如图 10-2 所示。



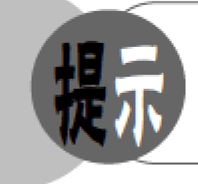
图 10-2　应用内部样式表效果

3. 导入外部样式表

导入外部样式表是指在内部样式表的<style>标记中使用@import 导入一个外部样式表，示例代码如下所示：

```
<style type="text/css" >
  @import "skin.css";
  </style>
```

示例中，使用@import 导入了样式表 skin.css。需要注意的是，使用时导入外部样式表路径，导入方法与连接外部样式表的方法一样。导入外部样式表相当于将样式表导入到内部样式表中，其输入方式更有优势。



导入外部样式表必须在样式表的开始部分，其他内部样式表之上。

例如，将代码 10.2 的 link 语句替换为代码 10.4 的内容，运行效果与原来相同。

代码 10.4　导入外部样式表

```
<style type="text/css" >
@import "Style1.css";
</style>
```

4. 内嵌样式表

内嵌样式表混合在 XHTML 标记里使用，通过这种方法，可以很简单地对某个元素单独定义样式。使用内嵌样式表的方法是直接在 XHTML 标记中使用 style 属性，该属性的内容就是 CSS 的属性和值，代码如下所示：

```
<body style="background-image:url("flower.jpg");background-position:
center">
  <h3 style="color:black">使用 CSS 内嵌样式</h3>
</body>
```



内嵌样式表只能对 HTML 标记定义样式，而不能使用类选择符或者 ID 选择符定义样式。

例如，在“外部样式 1.html”页面内 body 标记区的 h1 标记前添加代码 10.5 应用内嵌样式，最终得到的效果与图 10-2 相同。

代码 10.5　内嵌样式表

```
<div id="logo"style="background-image: url(chuangnei.gif);background-repeat:
no-repeat;height: 58px;">
</div>
```

上述介绍的 4 种引用 CSS 样式表方法可以混合使用，但根据优先权原则，引入样式表的方法不同应用也不同。其中，内嵌样式表优先权最高，接下来是连接外部样式表、内部样式表和导入外部样式表。

10.2　基本属性

从 CSS 的基本规则中可以看出，属性是 CSS 中非常重要的部分。在掌握如何使用选择符和 CSS 应用方式后，从本节开始将详细介绍 CSS 中属性的使用。掌握 CSS 的各种属性是编辑

页面样式表的前提条件，根据认知原则本节会首先介绍 CSS 中最基本的属性，包括字体、文本、背景和列表。

10.2.1　字体

字体属性是 CSS 中使用频率最高，也是最简单的样式属性。传统 XHTML 仅提供了对字体颜色、大小和类型的 3 种设置，而在 CSS 中可以对字体有更详细的设置，从而实现更加丰富的字体效果。

表 10-1 列出了 CSS 中用于控制字体的常用属性。

表 10-1　常用字体属性

属性	兼容性	说明
font-family	IE4+ , NS4+	指定使用的字体类型，可为此属性赋多值，系统将自动选择支持的字体显示
font-style	IE4+ , NS4+	指定字体显示的样式，取值为 normal、italic 或者 oblique
font-variant	IE4+ , NS6+	指定字体是否变形，取值为 normal 或者 small-caps
font-weight	IE4+ , NS4+	指定字体的加粗属性
font-size	IE4+ , NS4+	设置字体的大小，取值可以为绝对大小、相对大小、长度或者百分比
font	IE4+ , NS4+	指定字体的复合属性
letter-spacing	IE4+ , NS6+	指定字体之间的间隔大小
word-spacing	IE6+ , NS6+	指定单词之间的空白大小
line-height	IE4+ , NS4+	指定字体的行高，即字体最底端与字体内部顶端之间的距离

下面使用表 10-1 中的字体属性定义几种字体样式，在这些样式中综合使用了上述的属性，代码 10.6 所示为定义后的字体样式。

代码 10.6　定义字体样式

```
<style type= "text/css" >
.font1 {
      font-family: "华文行楷", "Courier New";
      font-size: 24px;
      font-weight: bold;
      color: #000;
}
.font2 {
      font-size: 16px;
      font-style: italic;
      font-variant: normal;
      font-weight: 600;
}
.font3 {
      font-style: oblique;
      color: #F00;
}
.font4 {
      letter-spacing: 0.5em;
```

```
    }
</style>
```

上述代码定义了4个样式规则，并在每个规则中都设置了字体属性。接下来创建“fontDemo.html”文件，然后在页面重要位置添加代码10.7所示内容，应用定义的字体样式。

代码 10.7　应用字体样式

```
<p class="font1">窗内网</p>
<p class="font2"> 域名: www.itZcn.com </p>
<p class="font3"> 成立日期: 2009-01-01</p>
<p class="font4"> 宗旨: 打造一流 IT 学习乐园 推进无纸化教学进程</p>
```

在浏览器中打开fontDemo.html，看到图10-3所示效果。这4种样式之间不会互相影响。

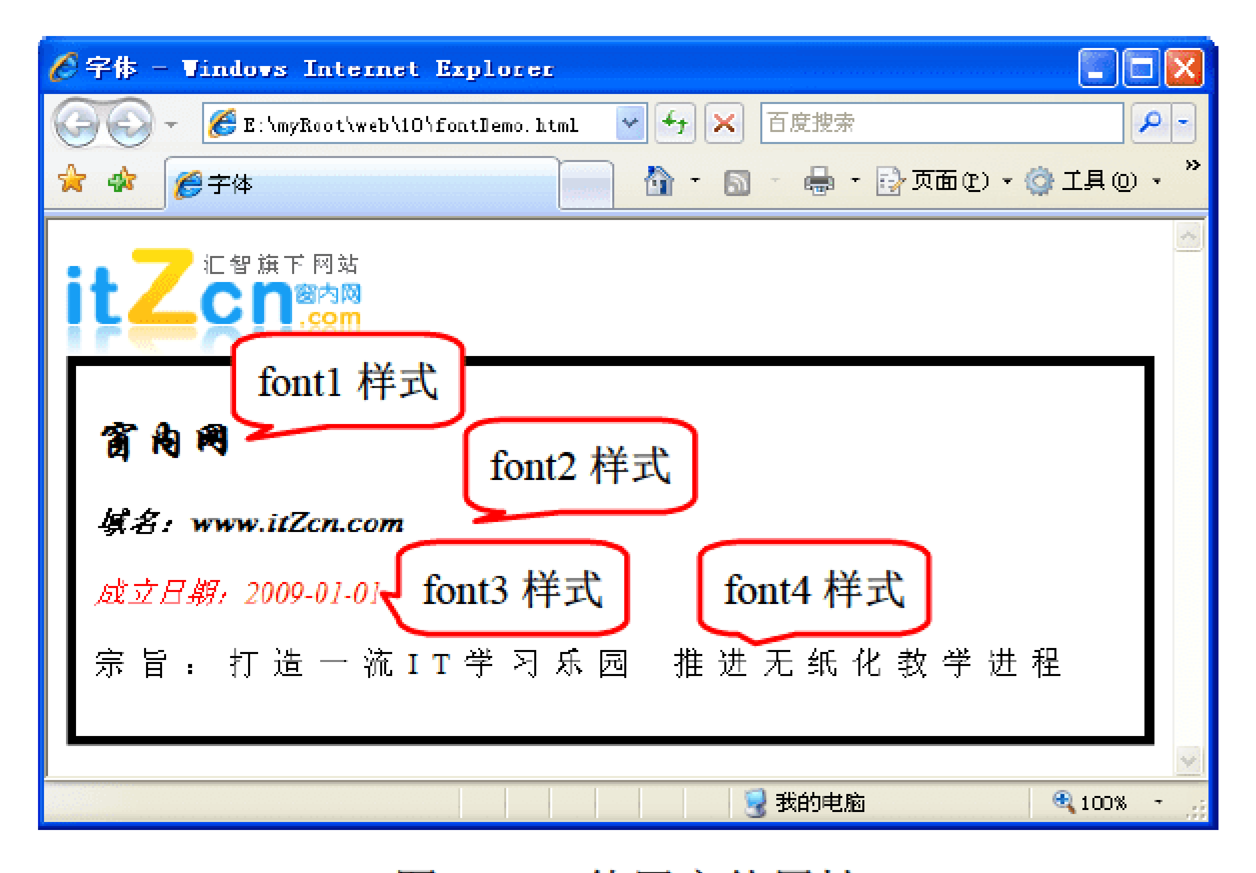


图 10-3　使用字体属性

10.2.2　文本

网页最主要的作用就是给用户提供信息内容，而这些信息通常是以文本方式展现的。文本的放置与效果的显示会直接影响到页面的布局及风格。在上节学习了构成文本的字体属性，本节将学习CSS提供的文本属性，从而实现对页面中文本的控制，表10-2列出了常用文本属性。

表 10-2　常用文本属性

属性	兼容性	说明
text-indent	IE4+ , NS4+	设置文本的缩进，默认值为 0
text-align	IE4+ , NS4+	设置文本的水平对齐方式，取值为 left、right、center 或者 justify
text-decoration	IE4+ , NS4+	设置文本的修饰属性，取值为 none、underline、blink、overline 或者 line-through
text-transform	IE4+ , NS4+	设置文本的大小写，取值为 none、capitalize、uppercase 或者 lowercase
vertical-align	IE4+ , NS4+	设置文本的垂直对齐方式
white-space	IE5.5+, NS4+	设置文本内对空白的处理方式
direction	IE5+	设置文本流入的方式，取值为 ltr、rtl 或者 inherit

下面以上节示例为基础，为 fontDemo.html 中字体样式添加文本修饰属性。首先为.font1 样式定义 text-align 属性，使其在页面中居中显示，代码如下所示：

```
.font1 {
    text-align: center;
}
```

接下来是.font2 样式，使用 uppercase 作为 text-transform 属性值，指定在文本中使用大写，再添加 text-indent 属性设置缩进 20px，这些样式代码如下：

```
.font2 {
      text-transform: uppercase;
      text-indent: 20px;
}
```

为.font3 样式添加 text-decoration 属性实现显示下划线，代码如下：

```
.font3 {
      text-decoration: underline;
}
```

再为.font4 样式添加如下代码，定义文本内容中不允许换行，并且使用删除线向右对齐：

```
.font4 {
      white-space: nowrap;
      text-decoration: line-through;
      text-align: right;
}
```

最后将上述样式代码依次添加到页面，并另存为“textDemo.html”，在浏览器中执行，效果如图 10-4 所示。



图 10-4 使用文本属性

10.2.3 背景

一个精美网页离不开背景的修饰，以及丰富的色彩。没有色彩的网页是空洞而没有生机

的，这就意味着一个优秀网页设计者不仅要能够合理安排页面布局，而且还要具有一定的色彩视觉和色彩搭配能力，这样才能够使网页更加精美也更具表现力，并给浏览者以亲切感。

现在，使用 CSS 的强大功能，能够为页面增加更多的色彩及背景选择，并且能够统一页面元素的色彩配置。表 10-3 列出了 CSS 中用于控制背景的常用属性。

表 10-3 背景属性

属性	兼容性	说明
color	IE4+ , NS4+	颜色取值可以是一个关键字或一个 RGB 格式的数字
background-color	IE4+ , NS4+	指定背景颜色，取值与 color 相同
background-image	IE4+ , NS4+	指定元素的背景图像，例如：p { background-image: url(top.jpg) }
background-repeat	IE4+ , NS4+	指定背景图像如何被重复，取值为 repeat、no-repeat、repeat-x 或者 repeat-y
background-attachment	IE4+ , NS6+	指定背景图像位置是随内容滚动，还是固定不动
background-position	IE4+ , NS6+	指定背景图像的最初位置，最常用值有 top、center、bottom、left 和 right
background	IE4+ , NS4+	设定背景的颜色、图像、重复、附件和位置

下面介绍一下如何使用表 10-3 中背景属性实现在页面中显示网站的 Logo 图片。首先使用 HTML 标记在页面中要显示 Logo 的位置进行定义，本例中使用 div 标记，定义语句如下：

```
<div id="logo"></div>
```

针对上述语句，应该使用 ID 选择符定义 CSS 样式，设置以背景形式显示图片 chuangnei.gif，定义背景不重复，再设置高度与图片高度相同，生成的 CSS 样式如下：

```
#logo {
    background-image: url(chuangnei.gif);
    background-repeat: no-repeat;
    height: 58px;
}
```

在页面中 Logo 图片一般仅出现一次，因此在上述示例中定义 background-repeat 属性的值为 no-repeat（不重复）。接下来介绍如何使用 background-repeat 属性重复背景显示。按照习惯先在页面中进行定义，语句如下：

```
<div id="sHeader">窗内网->新手入门->窗内网简介</div>
```

同样使用 ID 选择符定义所需的 CSS 样式，由于这里在 div 标记中使用了文本，因此还要添加 color 属性定义文本的颜色，最终 CSS 样式如下：

```
#sHeader {
    height:36px;
    color:#FF0000;
    background-image: url(bg.gif);
    background-repeat: repeat-x;
    line-height: 38px;
}
```

将上述代码添加到页面中，然后在浏览器中查看效果，如图 10-5 所示。



图 10-5 使用背景属性

10.2.4 列表

列表是页面中显示信息的一种常见方式。可以将相关具有并列关系的内容整齐地垂直排列，不仅很好地归纳了内容，而且也使页面显得整洁，增强页面的条理性。CSS 为控制列表提供了符号列表、图像列表和位置列表 3 种样式。表 10-4 列出了与列表相关的属性。

表 10-4 列表属性

属性	兼容性	说明
list-style	IE4+ , NS4+	设置列表的复合属性
list-style-image	IE4+ , NS6+	指定一个图像作为项目符号，例如 p{list-style-image:url(top.jpg)}
list-style-position	IE4+ , NS6+	指定项目符号与列表项的位置，取值为 outside 或者 inside
list-style-type	IE4+ , NS4+	设置列表中列表项使用的项目符号，默认为 disc，表示实心圆

首先在页面中添加代码 10.8，然后到浏览器中查看使用列表的默认效果，如图 10-6 所示。

代码 10.8 添加列表

```
<ul>
  <li>品质保证  优秀视频保障课程质量</li>
  <li>强化训练  实战项目提升动手能力</li>
  <li>专家答疑  学员问题有问必答</li>
  <li>学习灵活  在线充电，时间自己支配</li>
  <li>针对性强  为每个职业量身订做课程</li>
  <li>高时效性  针对主流编程和网站开发方面</li>
</ul>
```

通过 list-style-type 属性可以修改默认的项目符号，该属性有 9 个属性值，分别是：disc、circle、square、decimal、upper-alpha、lower-alpha、upper-roman、lower-roman 和 none，图 10-7 所示为几种常用项目符号的效果。

下面使用表 10-4 中列出的列表属性对默认效果进行修饰，设置一个列表项图片替换默认项目符号。在页面中添加所需的 CSS 样式，如代码 10.9 所示。

代码 10.9　自定义列表项目符号样式

```
li {
    color:#000;
    list-style-image: url(li.gif);
    list-style-type: none;
    list-style-position: outside;
}
```

此时再浏览则会看到图 10-8 所示自定义列表的项目符号效果。

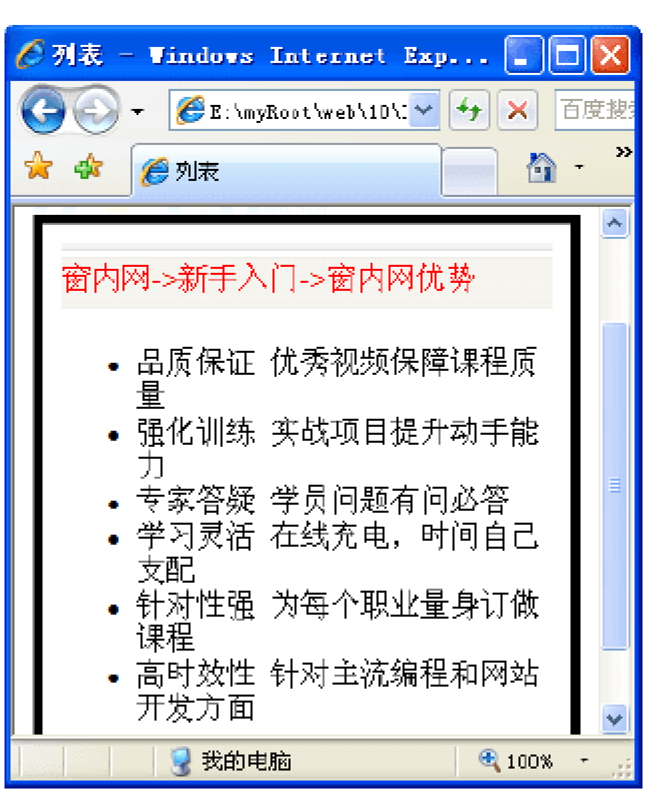


图 10-6　默认效果

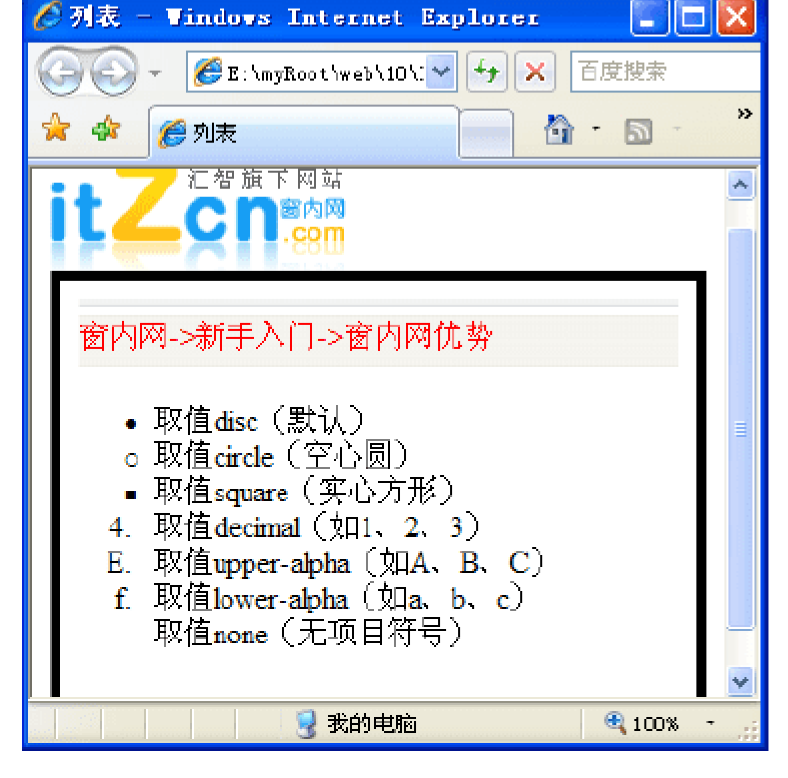


图 10-7　list-style-type 属性

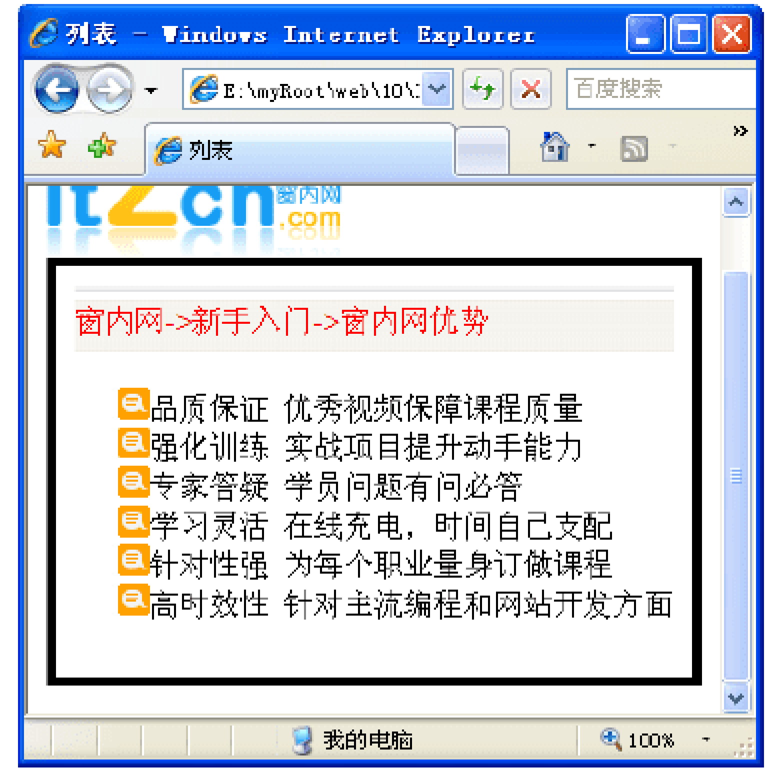


图 10-8　自定义项目符号

10.3　区块属性

在进行页面设计时，要保证页面元素出现在适当的位置，常常需要使用表格来完成。这是因为表格包含的边框能够为整个页面建立复杂的结构，而且还能使页面看起来更加美观、整洁。CSS 中的区块属性就提供了这样一种功能，能够为页面元素定义边框，并修饰内部间距，从而优化文本内容的显示效果。

10.3.1　区块模型

理解区块的概念是理解DIV+CSS制作页面的基础，它控制页面中元素的安排和显示方式。需要注意区块的实际应用以及绝对定位和相对定位，掌握如何控制页面中的每个元素。

通常在使用 CSS 设计页面布局时，所有页面元素都包含在一个矩形框内，称为元素框。元素框描述了元素及其在页面布局中所占的空间大小，因此元素框可以影响其他元素的位置及大小。例如，页面中第一个元素框为 10px，那么下一个元素框就处于离顶部 10px 距离的位置。如果第一个元素框增加为 20px，则下一个元素框就要再下移 10px。而整个页面就是由这些大大小小，但不会重叠的元素框组成的。

在设计页面布局时，网页设计者要充分考虑所有页面元素的边框与元素框之间边距的布

置，才能使页面紧凑，而又不失条理。要完全理解边距与间隙，设计者就必须清楚地明确各边间距及填充的位置，而边距、边框、填充及内容共同构成了一个区块，图 10-9 所示是示意图。

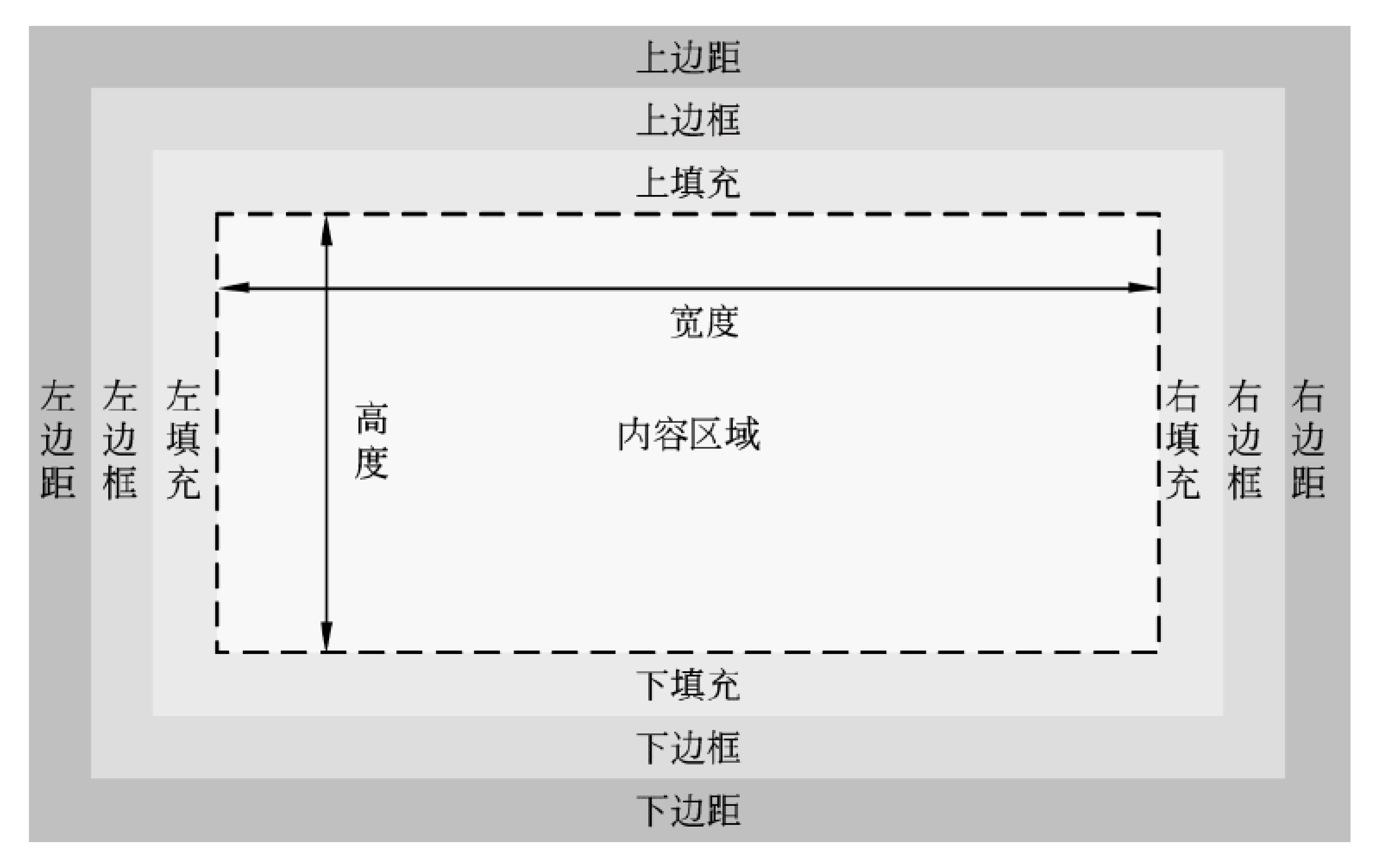


图 10-9　区块示意图

从示意图中可以清楚地看出边距与填充所控制的区域。其中元素框就是那个最深阴影的矩形框。这时，有的读者可能会问，边距和填充产生的作用都是定义距边界的距离，两者有什么区别呢？其实，边距和填充最大的区别就是背景的显示，例如对于如下的示例区分代码，运行效果如图 10-10 所示。

```
<p style="margin:30px; background-color:#179DF4; ">
  打造一流 IT 学习乐园　推进无纸化教学进程
</p>
<p style="margin:30px; padding:20px; background-color:#179DF4; ">
  打 造 一 流 IT 学 习 乐 园　 推 进 无 纸 化 教 学 进 程
</p>
```

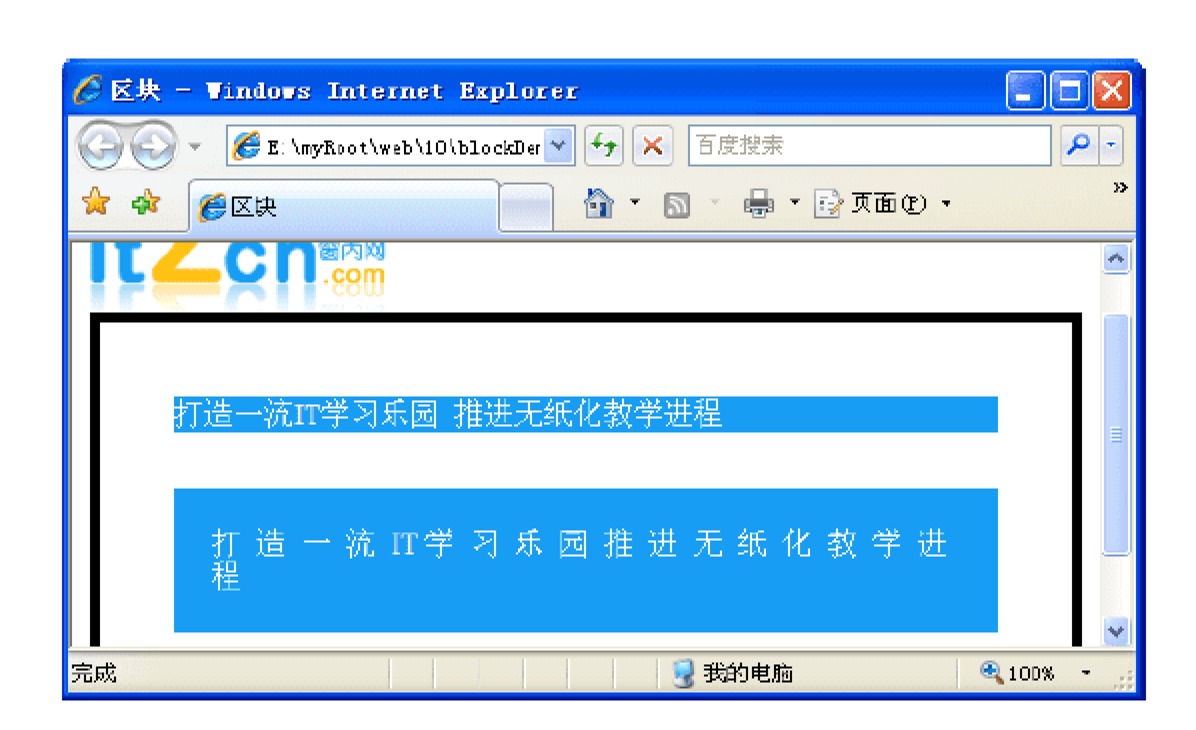


图 10-10　边距与填充的区别

从中可以看出，第一个段落只设定了边距，而背景色只显示在元素框区域，这说明背景

色不作用于填充。第二个段落设定了填充，而背景色除了显示在元素框区域，也覆盖了填充区域，这说明背景色作用于填充。

区块由内容、边距、填充以及边框共同构成，除内容外，其他都是可选的。这就会产生一个问题，例如对于 div 标记在 IE 6.0 中会有两像素的边距，而其他浏览器则没有。解决方法是通过将元素的 margin 和 padding 设置为 0 来覆盖这个浏览器的默认值。实现时，可以分别进行，也可以使用通用选择器对所有元素进行设置。下面给出的是使用通用选择器时的 CSS 样式：

```
*{
    margin:0;
    padding:0;
}
```

另外，在 CSS 中 width 和 height 属性指的是内容区域的宽度和高度。增加边框、边距和填充不会影响内容区域的大小，但是会增加区块的总大小。假设区块的每个边上有 30 像素的边距和 20 像素的填充，如果希望这个区块的宽度达到 500 像素，就需要将内容的宽度设置为 400 像素，CSS 样式如下：

```
#Page {
    width: 400px;
    margin: 30px;
    padding: 20px;
}
```

使用上述 CSS 样式后，Page 容器所定义的区块宽度即达到 500 像素，图 10-11 中给出上述 CSS 样式的作用示意图。

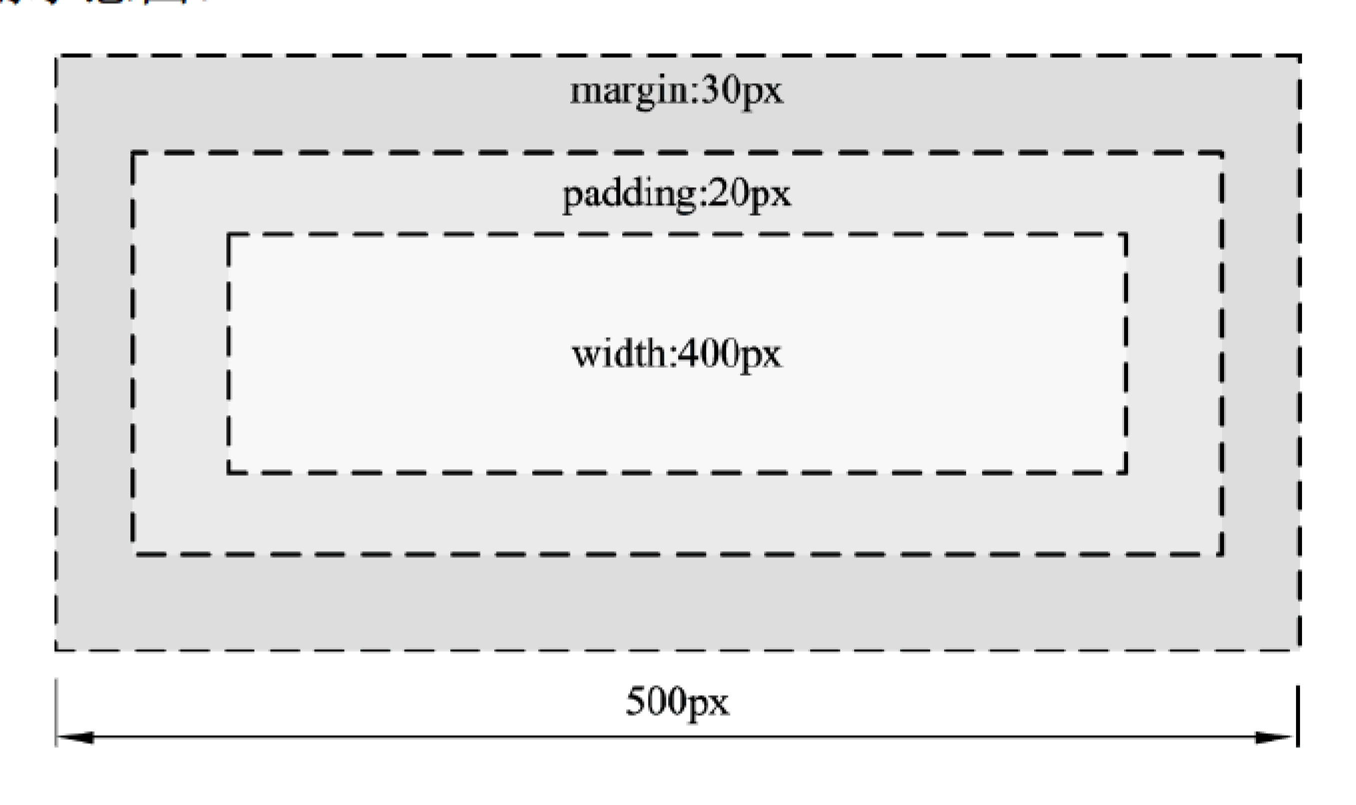


图 10-11　定义区块

10.3.2　边框

页面元素的边框就是将元素内容及间隙包含在其中的边线，类似于表格的外边线。每一个页面元素的边框可以从 3 个方面来描述：颜色、样式和宽度，这 3 个方面决定了边框所显

示出来的外观。CSS 中定义这 3 方面的属性如表 10-5 所示。

表 10-5　边框属性

属性	兼容性	说明
border	IE4+ , NS4+	设置边框样式的复合属性
border-color	IE4+ , NS6+	设置边框的颜色值
border-style	IE4+ , NS4+	设置边框的样式
border-width	IE4+ , NS4+	设置边框宽度
border-top 、 border-right 、 border-bottom 、 border-left	IE4+ , NS6+	分别设置上、右、下和左边框的复合属性
border-top-color、border-right-color、border-bottom-color、border-left-color	IE4+ , NS6+	分别设置上、右、下和左边框的颜色
border-top-style、border-right-style、border-bottom-style、border-left-style	IE4+ , NS6+	分别设置上、右、下和左边框的样式
border-top-width、border-right-width、border-bottom-width、border-left-width	IE4+ , NS4+	分别设置上、右、下和左边框的宽度

在使用边框样式时，border-style 属性是最常用的，使用其可选值可以定义边框的样式，包括：none（无边框）、dotted（点划线）、dashed（虚线）、solid（实线）、double（双线）、groove（槽状）、ridge（脊状）、inset（凹陷）和 outset（凸出）。代码 10.10 中列出了定义的 4 种边框样式代码。

代码 10.10　使用边框样式

```
.border1 {
    font-family: Arial, "黑体";
    font-size: 16px;
    font-weight: bold;
    height: 0px;
    border-top-width: 1px;
    border-bottom-width: 1px;
    border-top-style: solid;
    border-bottom-style: solid;
    border-top-color: #000;
    border-bottom-color: #000;
}
.border2 {
    font-family: Arial, "黑体";
    font-size: 16px;
    color: #000;
    width: 200px;
    padding-top: 5px;
    padding-bottom: 5px;
    border-top-width: 3px;
    border-bottom-width: 3px;
    border-top-style: double;
    border-bottom-style: double;
```

```
        border-top-color: #000;
        border-bottom-color: #000;
    }
    .border3 {
        font-family: Arial, "黑体";
        font-size: 16px;
        font-weight: bold;
        height: 30px;
        border: 1px dashed #FFF;
    }
    .border4 {
        font-family: Arial, "黑体";
        font-size: 16px;
        font-weight: bold;
        margin: 3px;
        height: 30px;
        width: 300px;
        border-top-style: outset;
        border-right-style: outset;
        border-bottom-style: outset;
        border-left-style: outset;
        border-top-color: #000;
        border-right-color: #000;
        border-bottom-color: #000;
        border-left-color: #000;
        color: #000;
    }
```

接下来创建一个新的 HTML 文件，在其中应用上述 4 个样式，运行效果如图 10-12 所示。

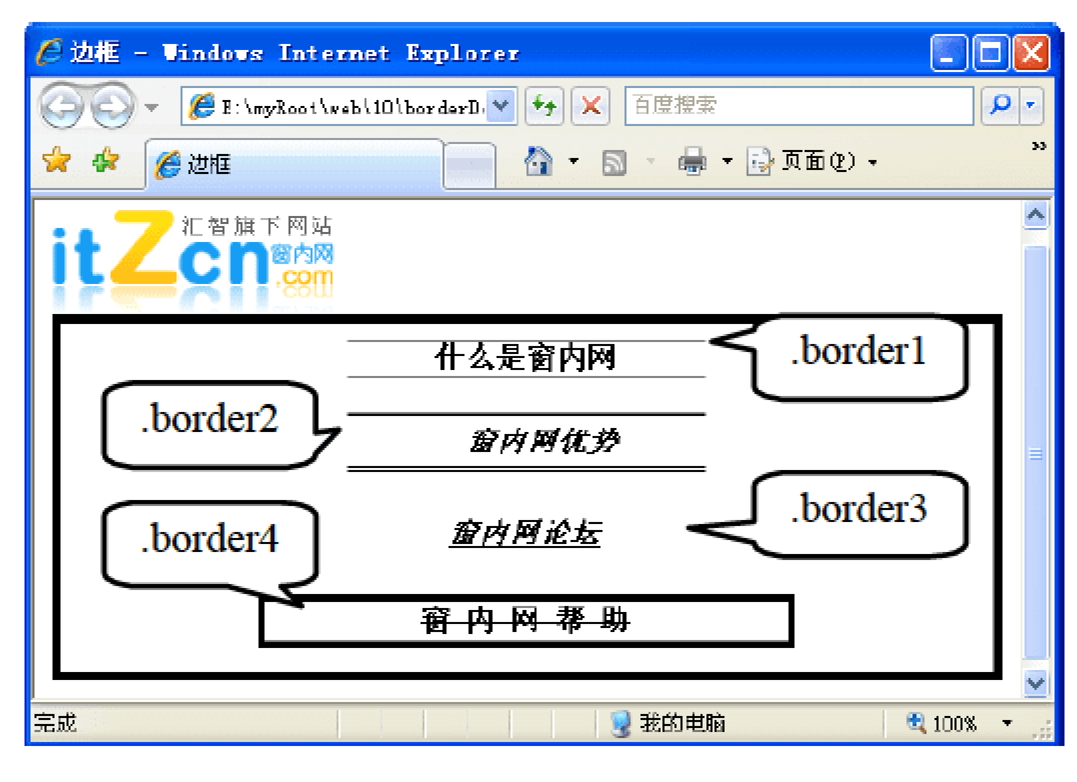


图 10-12　使用边框属性

10.3.3　间距

在 CSS 中使用 margin 属性定义元素四周的间距。该属性是个复合属性，也可以使用其包

含的 4 个子属性分别对上、右、下和左边的间距进行单独设置，如表 10-6 所示。

表 10-6 间距属性

属性	兼容性	说明
margin	IE4+ , NS4+	设置元素间距的复合属性，默认值为 0
margin-top	IE4+ , NS4+	设置元素上边的间距，默认值为 0
margin-right	IE4+ , NS4+	设置元素右边的间距，默认值为 0
margin-bottom	IE4+ , NS4+	设置元素下边的间距，默认值为 0
margin-left	IE4+ , NS4+	设置元素左边的间距，默认值为 0

各属性的属性值可以是一个确定的值，也可以是一个百分比，该百分比是相对于其上级元素的宽度（width）。如果要在 margin 复合属性中一次性地分别定义 4 个子属性的值，格式为：

```
margin: [margin-top] [margin-right] [margin-bottom] [margin-left]
```

需要注意的是，必须严格按照上述 4 个子属性的顺序进行设置，否则页面的布局将不会按照预计效果显示。但是这 4 个子属性值的单位却可以不是统一的，例如：

```
margin:1cm 15mm 20px 5cm;
```

如果上、下、左和右 4 个边距的值相等，那么可以只设定一个 margin 属性值，例如：

```
margin:15px;
```

下面通过一个示例演示如何使用 margin 及其子属性设置元素的间距。新建一个页面，使用 margin 复合属性定义一个上、下、左和右边都是 10px 间距，背景颜色为#CCC 的.margin1 样式，如下所示：

```
.margin1 {
    margin:10px;
    background-color: #CCC;
}
```

使用百分比方式定义一个.margin2 样式，设置文本内容以上级元素宽度的 5%为间距进行显示，背景颜色为#CCC，如下所示：

```
.margin2 {
    margin:5%;
    background-color: #CCC;
}
```

接下来在定义的.margin3 样式中使用 margin 子属性依次设置上、下、左和右边的间距，背景颜色为#CCC，如下所示：

```
.margin3 {
    margin-top:1em;
    margin-bottom:15px;
    margin-left:50mm;
```

```
    margin-right:2.5cm;
    background-color: #CCC;
}
```

在.margin4 样式中使用 margin 属性的完整格式定义间距，如下所示：

```
.margin4 {
    margin:1cm 15mm 20px 5cm;
    background-color: #CCC;
}
```

添加上述代码后，在页面的<body>标记区内使用如下的代码应用这 4 个样式：

```
<div class="margin1">订阅班级都需要什么？</div>
<div class="margin2">取消班级订阅是否会返还所扣积分？ </div>
<div class="margin3">忘记我的登录密码，怎么办？</div>
<div class="margin4">如何取消对班级的订阅？</div>
```

将页面保存为 marginDemo.html，在浏览器中查看效果如图 10-13 所示。

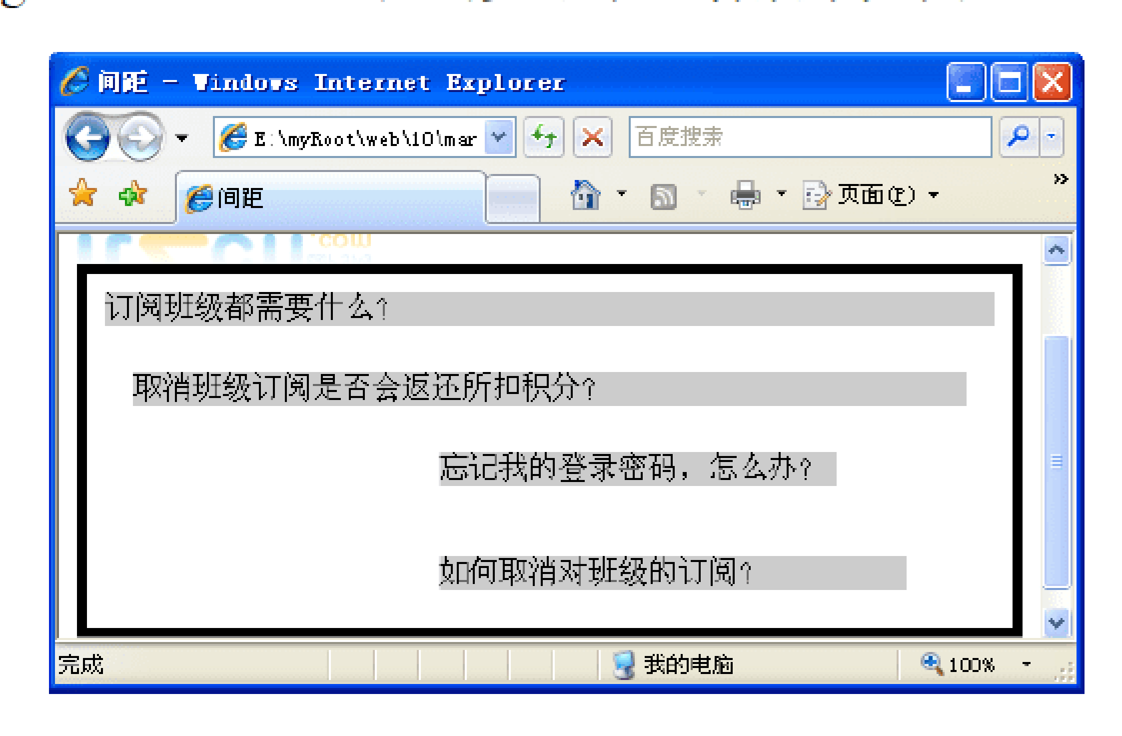


图 10-13　使用间距属性

10.3.4　填充

CSS 中控制元素填充的属性是 padding，该属性可设置元素与边框之间的填充宽度。与 margin 属性相似，padding 属性也是一个复合属性，包含 4 个子属性，分别是：padding-top、padding-right、padding-bottom 和 padding-left。

padding 属性值的设置方式与 margin 属性相似，其属性值可以是一个具体的长度，也可以是一个相对于上级元素的百分比。下面通过示例介绍 padding 属性及其 4 个子属性的应用及显示效果。

打开上节的 marginDemo.html 文件，然后将代码 10.11 中的填充属性样式依次添加到.margin1、.margin2、.margin3 和.margin4 中。

代码 10.11　填充属性

```
.margin1 {
```

```
        padding: 2px; /*使用复合属性的简写*/
    }
    .margin2 {   /*使用子属性*/
        padding-top: 20px;
        padding-right: 0px;
        padding-bottom: 0px;
        padding-left: 50px;
    }
    .margin3 {
        padding: 2mm;
    }
    .margin4 {
        padding: 2%; /*使用百分比*/
    }
```

将页面另存为 paddingDemo.html，再查看应用填充属性后的效果，如图 10-14 所示。



图 10-14　使用填充属性

10.4　位置属性

在 CSS 中定位与布局是密不可分的。定位可以将一个元素精确地放在页面上用户所指定的位置，而布局是将整个页面的元素内容整洁且完美地摆放。定位的实现是布局成功的前提。但是定位一直是 Web 标准应用中的难点，如果不能够清楚认识定位，可能会实现不了理想的布局效果。而如果清晰地掌握了定位的原理，就能够创建多种高级而精确的布局，并会让网页更加完美地实现。

10.4.1　定位

定位（Positioning）的原理其实很简单，就是使用有效、简单的方法精确地将元素定义到

页面的特定位置中。这个位置可以处于页面的绝对位置，也可以处于其上级元素中，还可以是到另一个元素或浏览器窗口的相对位置。

在 CSS 中实现页面定位，也就是定义页面中区块的位置，表 10-7 列出了 CSS 中全部的定位属性。

表 10-7　定位属性

属性	兼容性	说明
position	IE4+ , NS4+	定义元素在页面中的定位方式
left	IE4+ , NS4+	指定元素与最近一个具有定位设置的父对象左边的距离
right	IE5+	指定元素与最近一个具有定位设置的父对象右边的距离
top	IE4+ , NS4+	指定元素与最近一个具有定位设置的父对象上边的距离
bottom	IE5+	指定元素与最近一个具有定位设置的父对象下边的距离
z-index	IE4+ , NS6+	设置元素的层叠顺序，仅在 position 属性为 relative 或者 absolute 时有效
width	IE4+ , NS6+	设置元素框宽度
height	IE4+ , NS6+	设置元素框高度
overflow	IE4+ , NS6+	内容溢出控制
clip	IE4+ , NS6+	剪切

前 6 个属性是实际的定位属性，后面的 4 个是有关属性，用来设置区块，或对区块中的内容进行控制。其中 position 属性是最主要的定位属性，它既可以定义区块的绝对位置，又可以定义相对位置，而 left、right、top 和 bottom 只在 position 属性中使用才会发挥作用。至于 z-index 属性，用来标识元素的层叠位置，其属性值为整数（integer）。属性值越大，顺序就越靠前，能够叠加到属性值小的元素上，从而优先显示。

CSS 根据 position 属性提供的 4 个属性值来实现 4 种不同的定位类型，分别是：absolute（绝对定位）、relative（相对定位）、static（静态定位）和 fixed（固定定位），默认为 static。

1. 绝对定位

在绝对定位时，区块的显示位置将以浏览器左上角的 0 点为开始点，其偏移方向及距离将配合边偏移属性的设定进行定位，并且是浮动在正常区块之上的。定义了绝对定位的区块将固定于页面中的某个区域，而且不会随着页面变化而变化。

如果元素中还包含有边距和填充，则区块的定位是在区块总的区域内根据绝对定位的边偏移属性值而设定的，如图 10-15 所示。

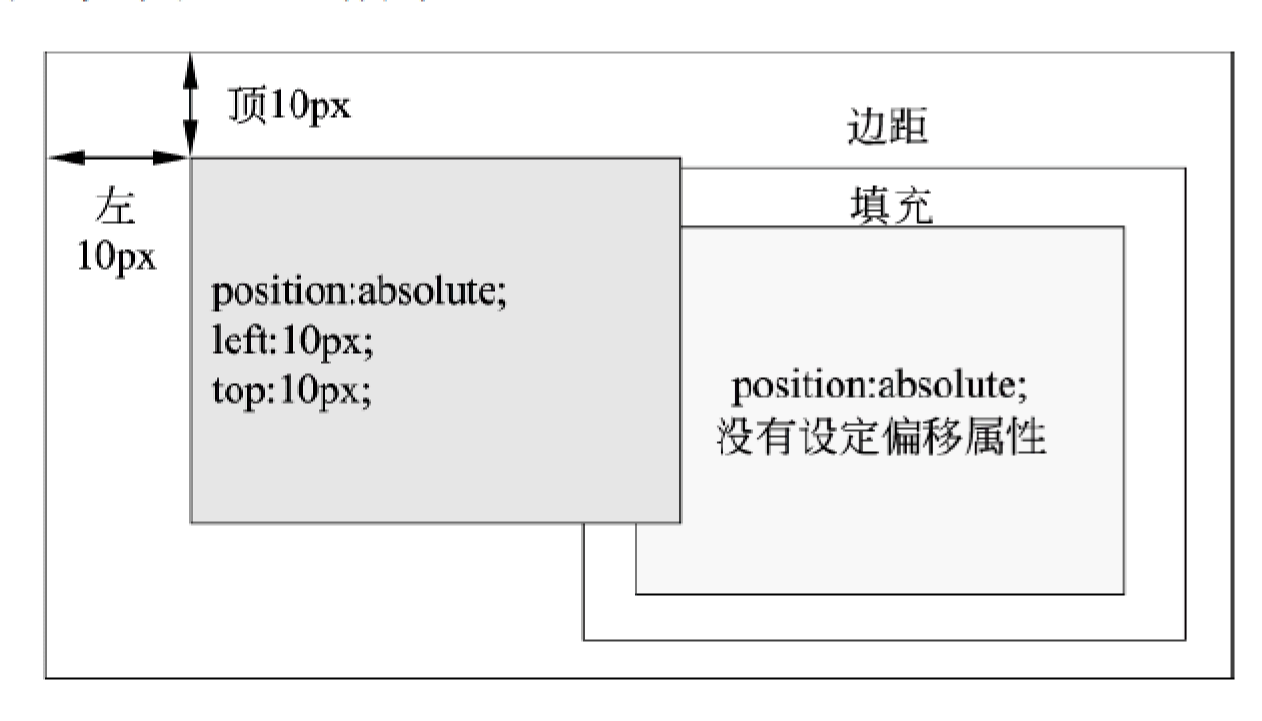


图 10-15　绝对定位示意图

图中左边的矩形块显示了包含边距和填充的区块应用偏移属性在页面中的绝对定位。参照图 10-15 显示，编写的测试示例如代码 10.12 所示。

代码 10.12　使用绝对定位

```
<title>绝对定位</title>
<style type="text/css">
#Box {
    width: 450px;
    height:200px;
    border: 1px solid #000;
    background-color: #FFF;
    padding:5px;
    position:absolute;
}
#Box1 {
    width: 450px;
    height:200px;
    padding:5px;
    border: 1px solid #000;
    background-color: #CCC;
    position:absolute;
    top:50px;
    left:50px;
}
</style>
<div id="Page">
<div id="Box">默认方式</div>
<div id="Box1">绝对定位 position:absolute</div>
</div>
```

在浏览器中查看区块在页面中的绝对定位，如图 10-16 所示。

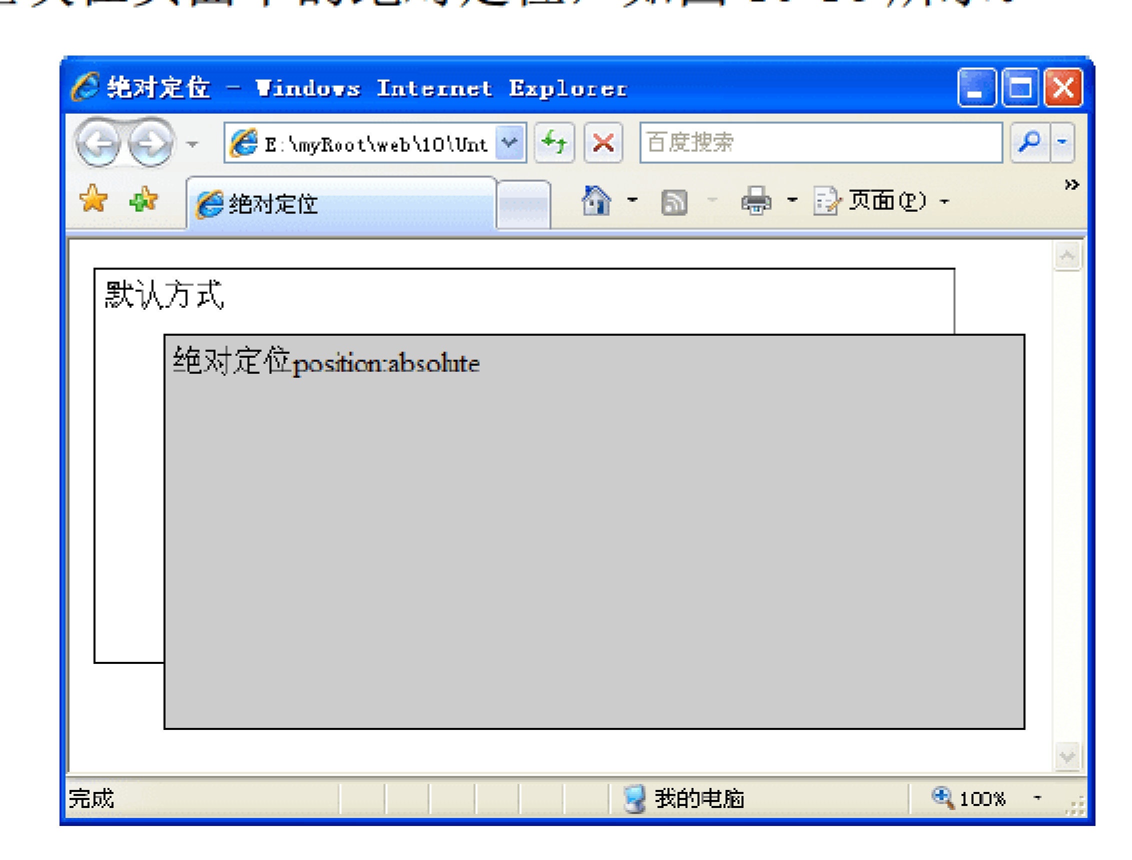


图 10-16　绝对定位

从图中可以看出，包含块的范围是其宽度与间距的和。但是绝对定位中设定偏移属性的元素块在显示时并不依赖于包含块的位置，而是直接由偏移属性值决定。没有设定偏移属性的元素块在显示时则处于默认位置。

使用绝对定位会产生一个问题。目前大多数网页都是居中显示的，而且元素与元素之间布局是紧密的。而绝对定位的开始位置是浏览器左上角的（0,0）坐标，当设定各元素块边偏移属性时，由于客户端屏幕分辨率的不同，各元素块的显示可能会有偏差。这是由于页面显示是随着分辨率的大小而自动适应的，而各元素块参照绝对定位的位置显示，那么在浏览器的视野范围内，原始页面可能超出屏幕或太小。

所以，优秀的页面设计是能够适用于各种屏幕分辨率，并且能够保证正常的显示的。要解决这个问题，在定位时最好使用相对定位。

2. 相对定位

相对定位是以上级区块的原始点为参照点，然后再配合偏移属性对区块进行定位。如果区块无上级元素，则以 body 为参照点；如果上级区块有边距或填充属性，则参照点以上级区块的内容区域为原始点进行定位，如图 10-17 所示。

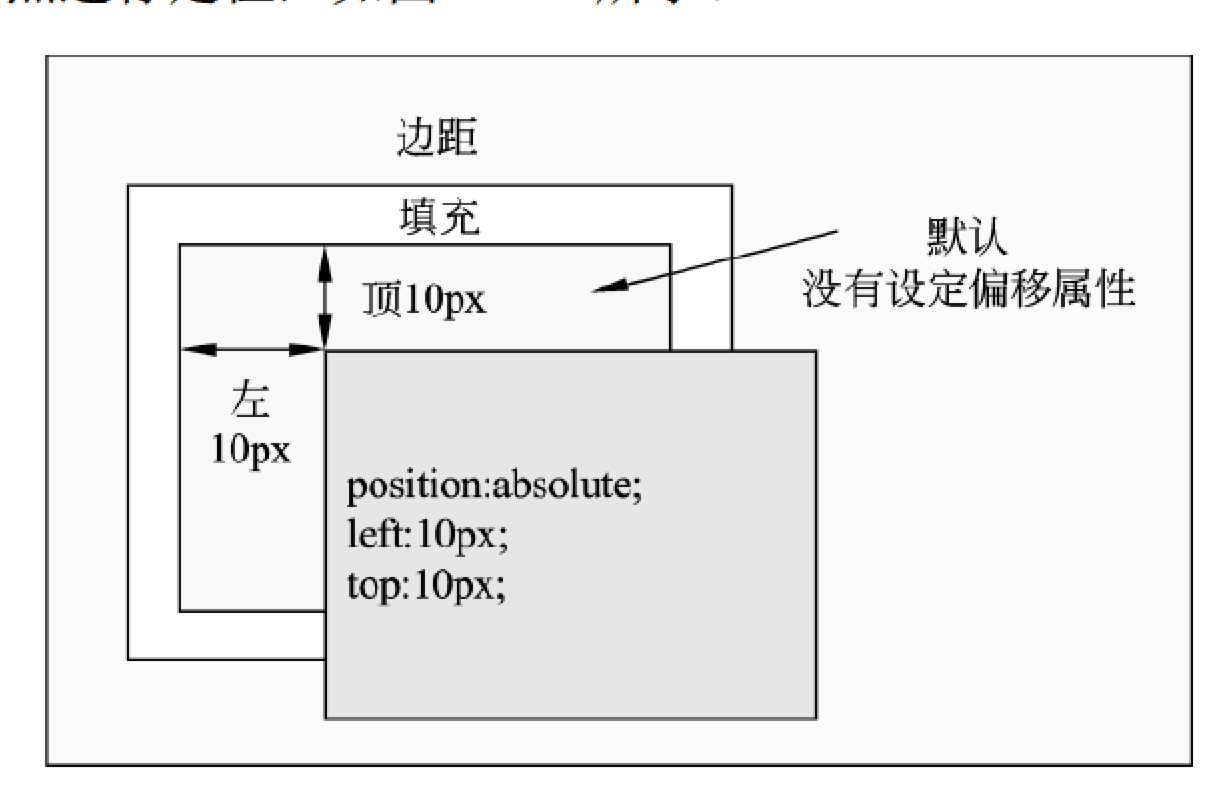


图 10-17　相对定位示意图

图中显示了设定偏移属性的元素块，在应用相对定位之后，是相对其上级元素的内容来进行定位的。例如在块标记中定位段落，使该段始终处于块标记的左边线和上边线 10px 的位置，如代码 10.13 所示。

代码 10.13　使用相对定位

```
<div style="margin-left:10px; background-color:#CCC; height:100px">
  <b>相对定位</b>
  <p style="background-color:#000;color:#FFF; position: relative; padding:
  10px;left: 10px;  top: 10px">
    以上级元素的原始点为参照点，然后再配合偏移属性对元素块进行定位。
  </p>
</div>
```

在浏览器中查看段落元素的定位，如图 10-18 所示。

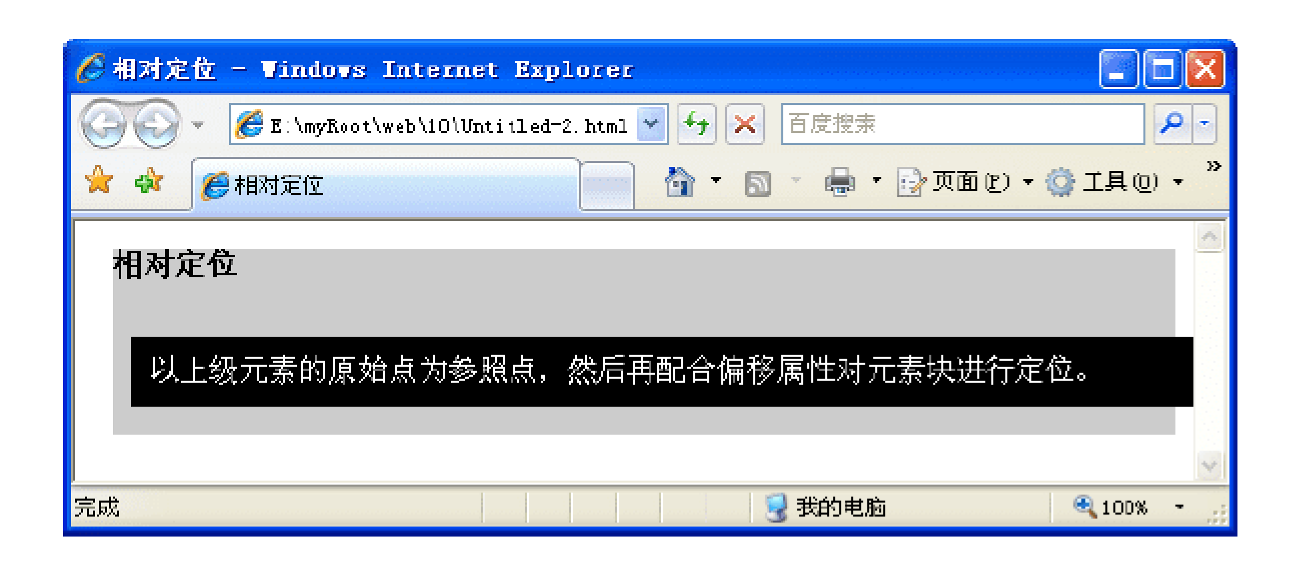


图 10-18　相对定位

图中，相对定位 4 个字所处的位置是块元素的内容区域，而段落元素块的位置要相对于该内容区域再向右及向下偏移 10px。而且无论块元素的位置如何移动，段落元素块在块元素内的位置始终不变。

3．固定定位

固定定位的参照位置不是上级区块而是浏览器窗口。所以可以使用固定定位来设定类似传统框架样式布局，以及广告框架或导航框架等。使用固定定位的元素可以脱离页面，无论页面如何滚动，始终处在页面的同一位置上。

4．静态定位

在未指定 position 属性值，或者使用 static 作为值时会静态定位（static）页面中的区块，在这种情况下区块会按照在页面中定义的先后顺序，按次序出现在页面中而不会发生偏移，也就是无特殊定位，区块遵循 HTML 的定位规则。

10.4.2　布局

当在页面中定义好区块的位置后，接下来的布局操作是如何安排、排列和改变区块的显示位置，同时设置页面中相邻两个区块的间距。表 10-8 中列出了 CSS 中全部的布局属性。

表 10-8　布局属性

属性	兼容性	说明
clear	IE4+，NS4+	指定元素中不允许有浮动对象的边，取值有 none（默认无）、left、right 和 both
float	IE4+，NS4+	指定元素是否浮动以及如何浮动，取值有 none（默认无）、left 和 right
clip	IE4+，NS6+	指定元素的可视区域，区域外的部分将隐藏
overflow	IE4+，NS6+	指定在元素的内容超过指定宽度和高度时的处理方式，默认为 invisible（隐藏）
overflow-x	IE4+	指定元素处理内容超过宽度时的方式
overflow-y	IE4+	指定元素处理内容超过高度时的方式
display	IE4+，NS4+	指定元素是否显示以及如何显示，默认为 inline（将元素以内联方式显示）
visibility	IE4+，NS6+	指定是否显示元素

这些主要是通过使用 float 属性完成，该属性的可选值有：none（无浮动）、left（向左浮动）和 right（向右浮动）。

使用 float 属性时要注意：定义 float 属性的元素会覆盖在其他元素上，而被覆盖的区域将

处于不可见状态。使用该属性能够实现内容环绕图片的混排效果，还可以在页面中实现多列布局。

首先介绍图文混排的实现。在页面中要显示内容的位置使用 div 进行标记，并设置 ID 为 content。向 div 内插入要实现环绕的图片，然后再输入一段要排列文字，如代码 10.14 所示。

代码 10.14　定义图片和文字

```
<div id="content">
<img src="mode.jpg" width="200" />
    <p>“师傅领进门，修行靠个人”，我们认为最好的师傅就是能够简明扼要地将基础的东西讲清楚，
    然后布置大量的练习甚至实战的项目逼迫您去不断地动手实践，最后能够在关键的地方帮助你一
    把，其他的就应该完全靠学习者自己去把握。</p>
    此处省略…
 </div>
```

下面使用 CSS 样式对页面进行布局，先定义正文中使用的字体大小及对齐方式，并使用 float 属性声明不允许正文（content）浮动，如代码 10.15 所示。

代码 10.15　设置文本样式

```
#content {
    float: none;
    text-align:left;
    font-size:16px;
}
```

在实现图文混排时，关键就是对图片设置排列方式，通过 float 属性的可选值可实现向左或向右排列。本例中制作向左排列的效果，CSS 样式如代码 10.16 所示。

代码 10.16　设置图片样式

```
#content img {
    float:left;
    padding: 5px;
    border: 2px solid #D6D6D6;
    margin: 10px;
}
```

将页面保存为 floatDemo.html，再查看效果如图 10-19 所示。将 float 的值设置为 right，将会实现图文向右环绕的效果，有兴趣的读者可以自己动手试一下。

float 属性的另一个重要作用是在 DIV+CSS 布局方式中实现多列。接下来介绍如何制作常见的两列和 3 列布局。

如果要使用浮动创建简单的两列布局，首先需要定义一个基本的 XHTML 页面框架。例如在下面的示例中将页面分为 Banner、Left、Main 和 Bottom。这 4 块包含在一个 id 是 Page 的 div 容器（标记）中，该 div 容器在页面水平居中。

示例 XHTML 页面的布局如代码 10.17 所示。



图 10-19　制作图文混排效果

代码 10.17　创建页面结构

```
<div id="Page">
  <div id="Banner">此处显示  id "Banner" 的内容</div>
  <div id="Left">此处显示  id "Left" 的内容</div>
  <div id="Main">此处显示  id "Main" 的内容</div>
  <div id="Bottom">此处显示  id "Bottom" 的内容</div>
</div>
```

要将这个 XHTML 页面的布局实现浮动，所需的 CSS 样式代码非常简单。只需为 div 容器定义宽度，并将 Left 容器设置向左浮动，Main 容器设置向右浮动。

代码 10.18 所示是让页面和 Page 容器居中的 CSS 样式。

代码 10.18　页面居中样式

```
body {
    text-align:center;
}
#Page {
    width: 600px;
    margin:0 auto;
    text-align:left;
}
```

下面分别针对 4 个块定义 CSS 样式，Banner 容器在顶部用于定义页面的导航、广告和 Logo 等，其宽度与页面相同。Left 容器在页面中作为左侧栏显示，可定义与页面相关的分类、信息聚合和公告等，宽度为 200 像素。接下来的 Main 容器在页面中占主要区域，定义要显示的详细内容，宽度为 375 像素。最后的 Bottom 容器作为页面的底部块，定义版权信息。代码 10.19 所示为根据这些需要定义的 CSS 样式。

代码 10.19　定义布局的显示样式

```
#Banner {  /*定义 ID 为 Banner 的样式*/
    height: 40px;
    border: 1px solid #000;
    margin: 5px;
    background-color: #CCC;
}
#Left {  /*定义 ID 为 Left 的样式*/
    width: 200px;
    float: left;
    margin: 5px;
    border: 1px solid #000;
    height: 100px;
    background-color: #CCC;
}
#Main {  /*定义 ID 为 Main 的样式*/
    margin: 5px;
    float: right;
    border: 1px solid #000;
    width: 375px;
    height: 100px;
    background-color: #CCC;
}
#Bottom {  /*定义 ID 为 Bottom 的样式*/
    border: 1px solid #000;
    margin: 5px;
    clear: both;
    height: 40px;
    background-color: #CCC;
}
```

这样就完成了使用 CSS 对两列页面进行布局的设计，运行效果如图 10-20 所示。

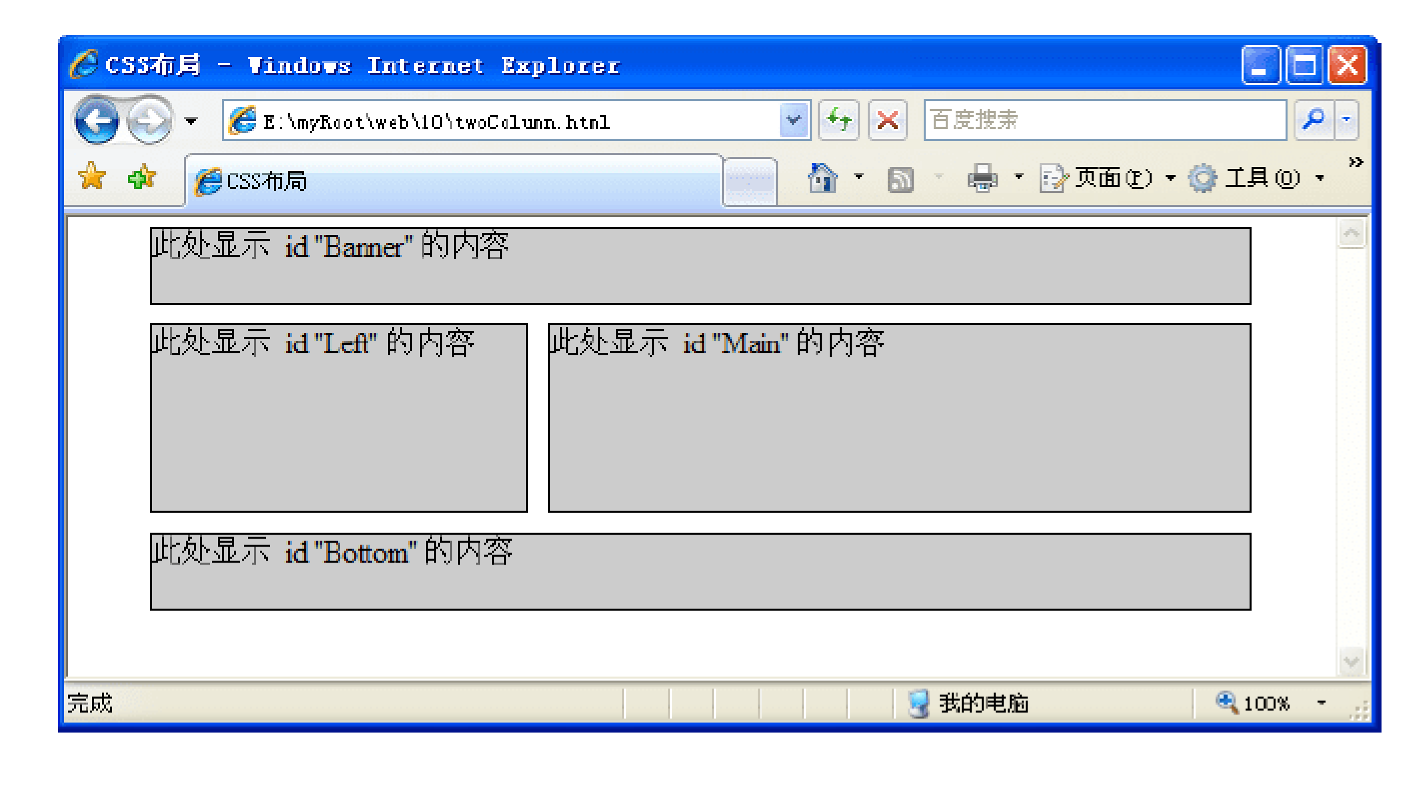


图 10-20　两列布局

在两列布局的基础上创建 3 列的浮动布局是非常简单的。以图 10-20 为例，所要做的就是在 Main 容器中添加两个新的 DIV 容器，如代码 10.20 所示。

代码 10.20　修改页面结构

```
<div id="Main">此处显示  id "Main" 的内容
  <div id="Main_Left">此处显示  id "Main_Left" 的内容</div>
  <div id="Main_Right">此处显示  id "Main_Right" 的内容</div>
</div>
```

有了页面布局结构，现在使用与两列布局技术相同的 CSS，在已经浮动的 Main 容器内将 Main_Left 和 Main_Right 容器分别向左和右浮动。这部分 CSS 样式如代码 10.21 所示。

代码 10.21　定义浮动

```
#Main_Left {  /*定义 ID 为 Main_Left 的样式*/
    width: 220px;
    border: 1px solid #000;
    background-color: #FFF;
    float: left;
    margin: 5px;
    height: 60px;
}
#Main_Right {  /*定义 ID 为 Main_Right 的样式*/
    float: right;
    width: 130px;
    background-color: #FFF;
    border: 1px solid #000;
    margin: 5px;
    height: 60px;
}
```

通过前面两步添加布局和 CSS 样式后，就形成了一个 3 列布局，效果如图 10-21 所示。

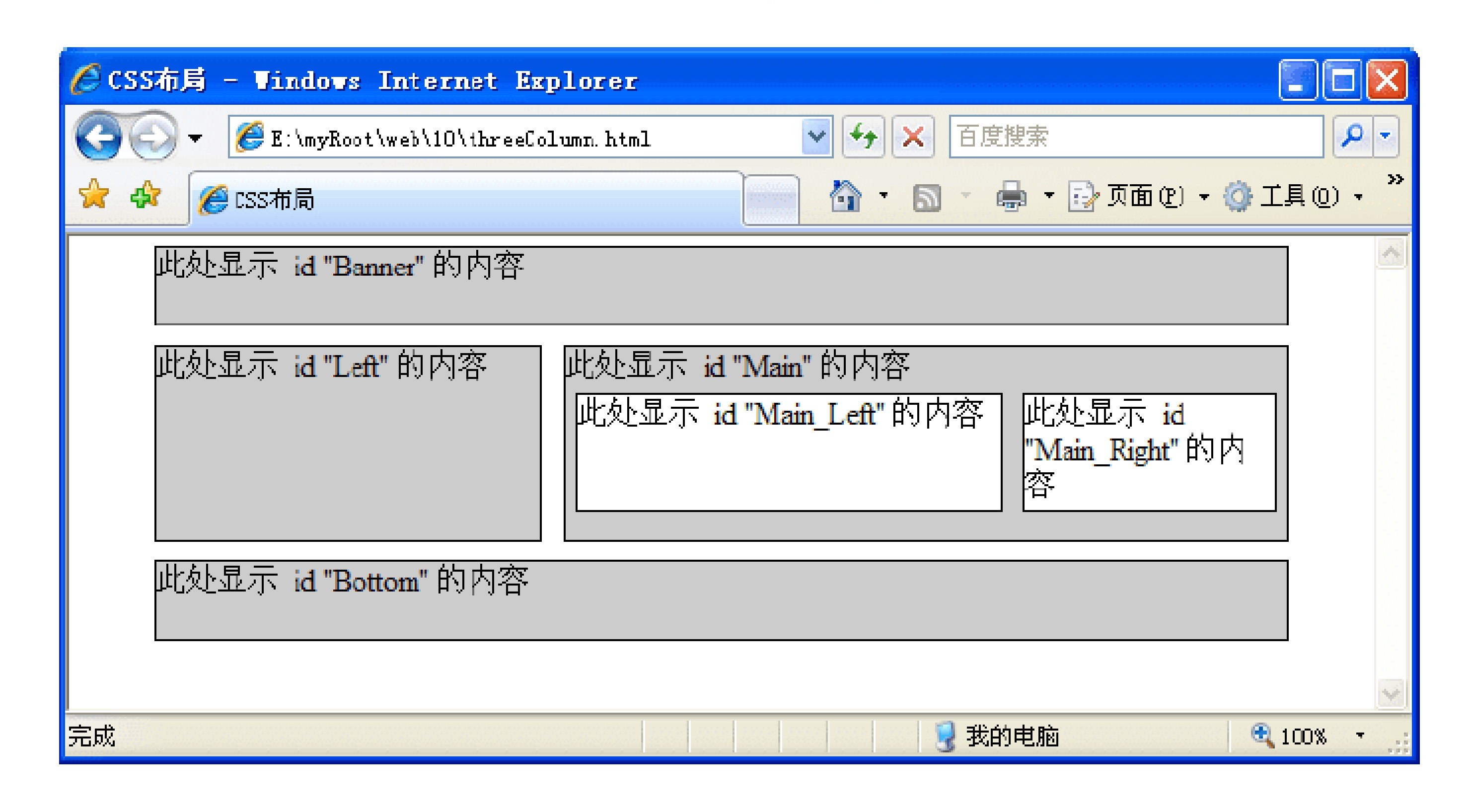


图 10-21　3 列布局

10.4.3 浮动模型

浮动模型是在设计页面布局时常用的一种模型。使用 float 属性定义的浮动框可以向左右浮动，直到其外边框遇到包含框或者另一个框的边框。此时因为浮动框不在文档的普通流中，所以文档流中的区块表现得就像浮动框不存在一样。

例如，有一个普通的 HTML 文档，其内容如代码 10.22 所示。

代码 10.22　HTML 文档

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style type="text/css">
#Page {
    width: 600px;
    margin:0 auto;
    text-align:left;
    background-color: #000;
}
#Box1{
    width: 150px;
    height:100px;
    background-color: #CCC;
    margin:10px;
}
#Box2{
    width: 150px;
    height:100px;
    background-color: #CCC;
    margin:10px;
}
#Box3{
    width: 150px;
    height:100px;
    background-color: #CCC;
    margin:10px;
}
</style>
</head>
<body>
<div id="Page">
<div id="Box1"> Box1 的内容</div>
```

```
    <div id="Box2"> Box2 的内容</div>
    <div id="Box3"> Box3 的内容</div>
    </div>
    </body>
    </html>
```


如代码中所示，在页面的 Page 容器中包含了 Box1、Box2 和 Box3 的内容，其运行效果如图 10-22 所示。

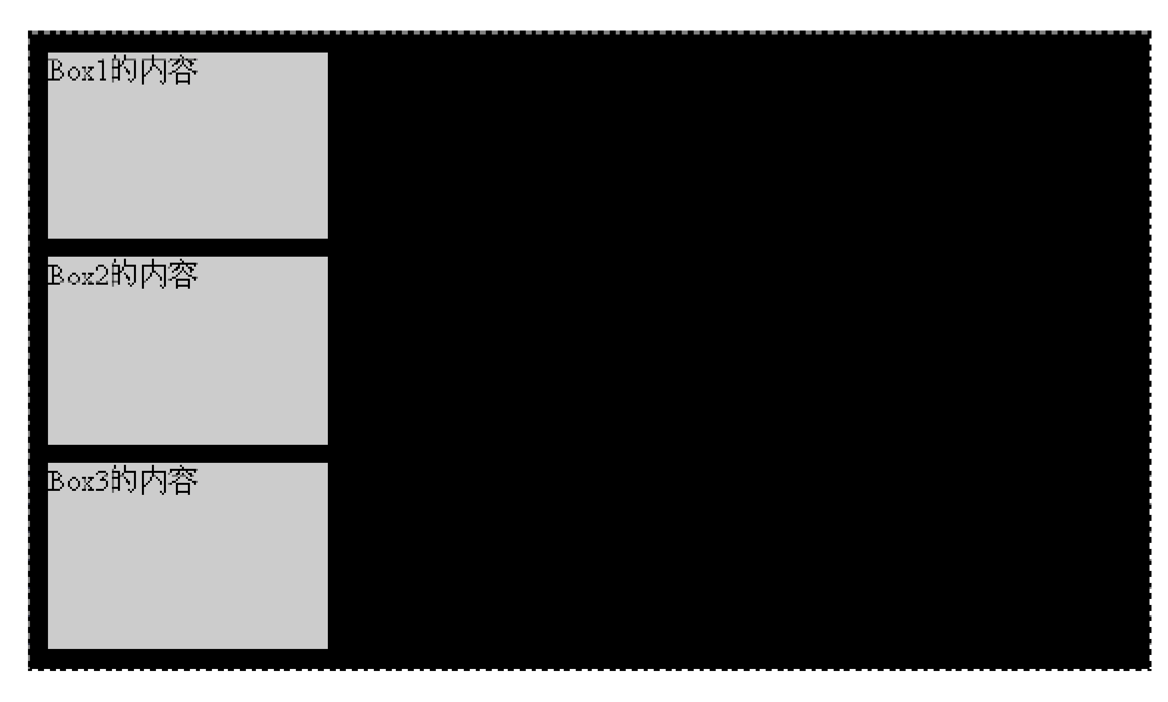


图 10-22　页面效果

现在，如果要将 Box1 向右浮动，此时它将脱离文档流并向右移动，直到其右边框碰到包含 Box1 的右边框（Page）。方法是在#Box1 中添加“float:right”，修改后的页面效果如图 10-23 所示。



图 10-23　Box1 向右浮动

如果要将 Box1 向左浮动，它将脱离文档流并且向左移动，直到其左边框碰到包含 Box1 的左边框（Page）。此时，由于它不再处于文档流中，所以将不会再占据页面空间，实际上覆盖了 Box2，使得 Box2 从页面中消失，如图 10-24 所示。



图 10-24　Box1 向左浮动

如果将 Box1、Box2 和 Box3 都修改为向左浮动，那么 Box1 向左浮动直到碰到 Page 的左

边框，另外两个框向左浮动直到碰到前一个浮动框，效果如图 10-25 所示。

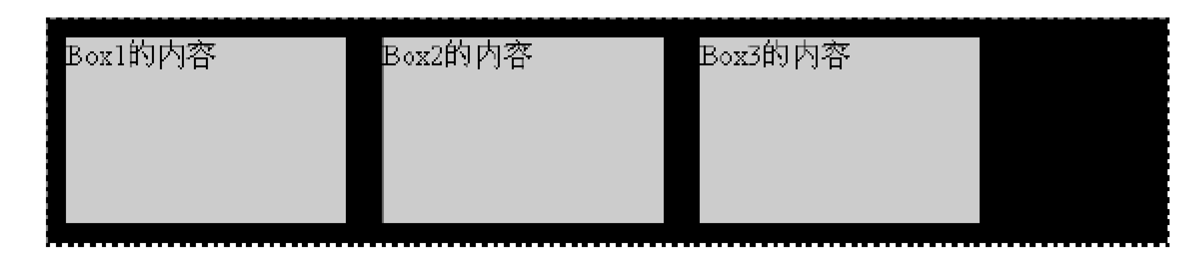


图 10-25　均向左浮动

此时，如果包含框（Page）的宽度太窄，无法容纳水平排列的 3 个浮动元素，那么其他浮动块将向下移动，直到有足够空间的地方，如图 10-26 所示。如果浮动元素的高度不同，那么当它们向下移动时可能会被其他浮动元素卡住，如图 10-27 所示。

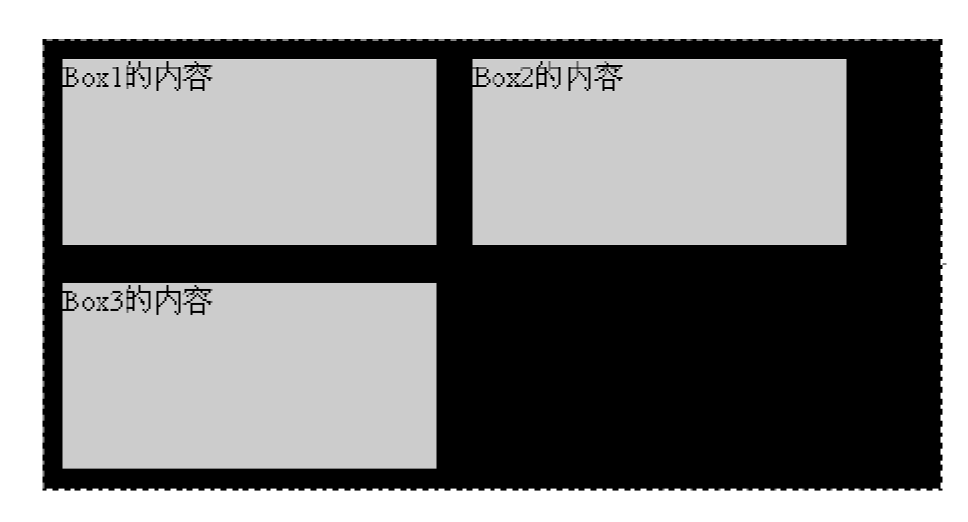


图 10-26　宽度不足时

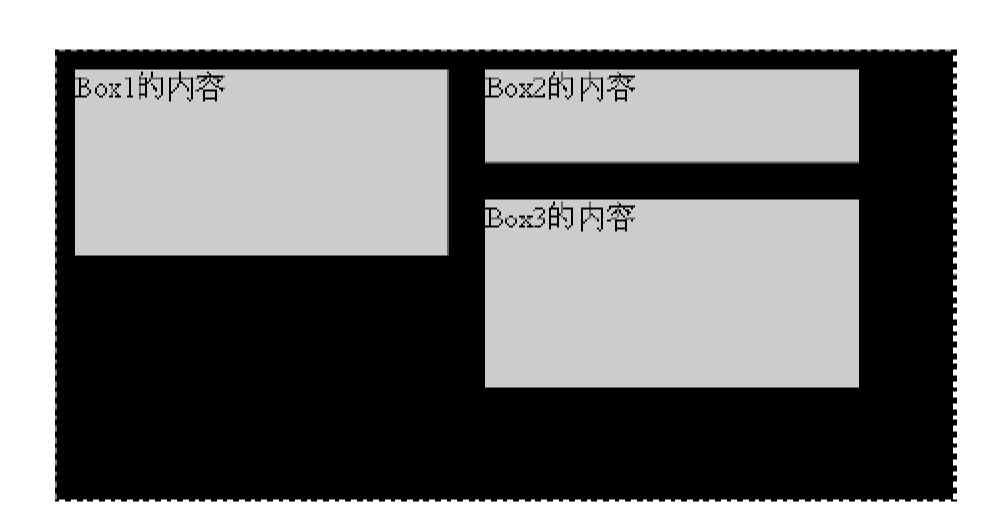


图 10-27　高度不同时

10.5　其他属性

除了本节之前介绍的 CSS 在字体、文本、列表、边框和布局方面的应用外，CSS 提供的功能还有很多，像单位、鼠标指针和滤镜等。单位的使用为 CSS 提供了除设置数字值之外的另一种选择，而适当地使用鼠标指针和滤镜可以制作出更加个性化的网页。

10.5.1　单位

为了使网页的页面布局合理，就必须精确地安排各页面元素的位置，而且还必须使页面的颜色搭配协调，以及字体的大小、格式规范，这些都离不开 CSS 中用来设置基础样式的属性。这些属性的基础是单位，有了这些单位才能够精准地布局各页面元素。本节将对进行网页设计时最常用的颜色和长度单位进行介绍。

1. 颜色

颜色主要用于对字体以及背景进行设置，还可以对区块和边框的颜色进行定义。在 CSS 中可使用 3 种方式来表示颜色单位，如表 10-9 所示。

表 10-9　颜色单位

属性	兼容性	说明
#RRGGBB	IE4+ , NS4+	3 个两位十六进制正整数，取值范围为：00~FF。例如#FF0000 表示红色
rgb(R,G,B)	IE4+ , NS4+	使用红、绿、蓝的正整数或百分数数值表示颜色。例如 rgb(255,0,0)表示红色
color name	IE4+ , NS4+	使用颜色名称表示颜色。例如 red 表示红色

2. 长度

在浏览器中要完全显示页面元素，又要布局合理，这需要设定元素间的间距，及元素本身的边界等，这都离不开长度单位的使用。CSS 中，长度单位可以被分为两类：绝对单位和相对单位，如表 10-10 所示。

表 10-10　长度单位

属性		兼容性	说明
相对	em	IE4+ , NS4+	相对于当前对象的字体大小
长度	ex	IE4+ , NS4+	相对于字符"x"的高度，通常为字体高度的一半
单位	px	IE3+ , NS4+	使用像素作为长度单位
	pt	IE3+ , NS4+	使用绝对的点（Point）作为长度单位
绝对	pc	IE3+ , NS4+	使用派卡（Pica）作为长度单位
长度	in	IE3+ , NS4+	使用英寸（Inch）作为长度单位
单位	cm	IE3+ , NS4+	使用厘米（Centimeter）作为长度单位
	mm	IE3+ , NS4+	使用毫米（Millimeter）作为长度单位

10.5.2　鼠标指针

经常使用操作系统的人们都知道，当将鼠标移动到不同的地方，或执行不同的操作时，鼠标的样式是不同的。比如当需要伸缩窗口时，将鼠标放置在窗口边沿处，鼠标会变成双向箭头状↔；当系统繁忙时，鼠标会变成漏斗状⌛。

CSS 不仅能够准确地控制及美化页面，而且还能定义鼠标指针的样式。当鼠标移动到不同的元素对象上面时，鼠标以不同的形状或图像显示。CSS 通过改变 cursor 属性（鼠标指针属性）来实现对鼠标样式的控制。

cursor 属性的兼容性为“IE4.0+ NS6.0+”，表 10-11 中列出了常用的 cursor 属性值及对应的指针样式。此外，在 cursor 属性中还能够通过 url 的链接地址自定义鼠标指针。

表 10-11　指针样式

属性值	说明	指针样式
auto	自动，按照默认状态自行改变	
crosshair	精确定位十字	
default	默认鼠标指针	
hand	手形	
move	移动	
help	帮助	
wait	等待	
text	文本	
n-resize	箭头上下双向	
e-resize	箭头左右双向	
ne-resize	箭头左下右上双向	
se-resize	箭头左上右下双向	
pointer	与 hand 相同	
url (cur/mouse.cur)	自定义鼠标指针	

10.5.3 滤镜

从 Internet Explorer 4.0 开始，浏览器便开始支持滤镜特效，允许用户使用简单的代码对文本和图片进行特殊处理。像实现模糊、彩色阴影、火焰效果、图片倒置、色彩渐变、风吹效果和光晕效果等。将滤镜、渐变和网页脚本语言结合起来，就可以建立一个动态交互的网页。

除了使用网页脚本语言实现滤镜特效，CSS 也在控制页面布局的同时提供了多种内置的滤镜特效。使用这种技术可以可视化地将滤镜和转换效果添加到页面元素上。CSS 的滤镜属性标识符是 filter，语法格式如下：

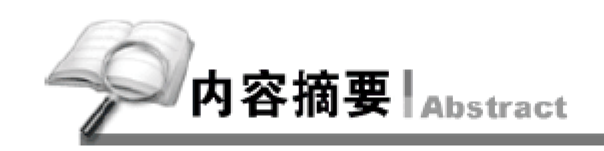
```
filter:filtername(parameters);
```

filtername 是要使用的滤镜名称，如 Alpha 和 DropShadow 等，表 10-12 列出了几种常用的滤镜名称。parameters 指定了该滤镜中的各参数，通过这些参数才能够决定滤镜显示的效果。

表 10-12　CSS 滤镜

滤镜名称	效果描述
Alpha	设置透明度
BlendTrans	实现图像之间的淡入和淡出的效果
Blur	建立模糊效果
Chroma	设置对象中指定的颜色为透明色
DropShadow	建立阴影效果
FlipH	将元素水平翻转
FlipV	将元素垂直翻转
Glow	建立外发光效果
Gray	灰度显示图像，即显示为黑白图像
Invert	图像反相，包括色彩、饱和度和亮度值，类似底片效果
Light	设置光源效果
Mask	建立透明遮罩
RevealTrans	建立切换效果
Shadow	建立另一种阴影效果
Wave	波纹效果
Xray	显现图片的轮廓，类似于 X 光片效果

第11章 JavaScript



内容摘要 | Abstract

JavaScript 是一种解释性的、基于对象和事件驱动并具有安全性能的脚本语言，主要应用于客户端，同时也是 Ajax 核心技术之一。由于 HTML 网页的互动性能比较弱，所以可以使用 JavaScript 开发交互式的 Web 网页。JavaScript 的用途很多，例如验证表单字段、动态更改页面元素外观、相应地调整页面以及在不离开本页的情况下，与服务器应用程序进行连接等。

本章首先对 JavaScript 语言进行简要概述，然后介绍 JavaScript 的基础知识，接着对控制流程、事件驱动、内置对象以及自定义对象进行详细的介绍。

学习目标 | Objective

- 了解 JavaScript 语言
- 理解并掌握 JavaScript 基础语法
- 理解并掌握 JavaScript 的控制语句
- 掌握如何定义和调用函数
- 理解基于对象的函数
- 熟悉 JavaScript 的系统函数
- 掌握并理解 JavaScript 的事件驱动
- 掌握并理解 JavaScript 的内置对象
- 熟悉 JavaScript 的浏览器对象
- 熟悉并理解 JavaScript 的自定义对象

11.1 JavaScript 语言概述

JavaScript 语言的前身称作 Livescript。自从 Sun 公司推出著名的 Java 语言之后，Netscape 公司引进 Sun 公司有关 Java 的程序概念，将原有的 Livescript 重新进行设计，并改名为 JavaScript。JavaScript 是描述性语言，可以被嵌入 HTML 文件中。在 HTML 基础上，使用 JavaScript 可以开发交互式 Web 网页。由于 JavaScript 在客户端上执行，所以提高了网页的浏览速度和交互能力。JavaScript 具有以下优点。

□ **简单性**

JavaScript 是脚本编写语言，采用小程序段的方式实现编程，像其他脚本语言一样，

JavaScript 同样也是解释性语言，提供一个简单的开发过程。其基本构成形式与 C、C++、VB 和 Delphi 十分类似。但不像这些语言一样需要先编译，而是在程序运行过程中逐行解释。与 HTML 标识结合在一起，从而方便用户使用操作。

- **动态性**

JavaScript 具有动态性，可以直接对用户输入做出响应，无需经过 Web 服务程序。对用户的响应，采用事件驱动的方式进行。在页面中执行某种操作所产生的动作，就称为“事件”。比如单击鼠标、移动窗口和选择菜单等都可以视为事件。事件发生后，会引起相应的事件响应。

- **跨平台性**

JavaScript 依赖于浏览器本身，与操作环境无关。只要计算机能运行浏览器，并且浏览器支持 JavaScript 就可以正确执行。

- **基于对象的语言**

JavaScript 是一种基于对象的语言，同时也可以看作一种面向对象的语言。这意味着它可以自定义对象。

- **安全性**

JavaScript 是一种安全性语言，不允许访问本地的硬盘，并且不能将数据存入到服务器上，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互，从而有效地防止数据的丢失。

- **节省 CGI 的交互时间**

CGI（Common Gateway Interface）是公共网关接口。随着 WWW 的迅速发展，有许多 WWW 服务器提供服务与浏览者进行交流、确认浏览者的身份等。通常由 CGI/PERL 编写响应的接口程序与用户进行交互。然而，通过网络与用户交互的过程中，增大了网络的通信量，并占用服务器的资源（如 CPU 服务、内存耗费等）。如果用户填表出现错误，交互服务占用的时间就会相应增加。被访问的热点主机与用户交互越多，对服务器的性能影响就越大。

11.2 基础语法

JavaScript 是一种脚本语言。脚本语言就是其生成的文件不能独立运行，必须依赖于一种平台。JavaScript 将代码直接写到 HTML 文档中，当浏览器读取这些代码时，则进行解释、执行。作为一种脚本语言，JavaScript 也有其自身的语法结构。在本节中主要介绍 JavaScript 的基础知识，包括变量、运算符和数据类型。

11.2.1 变量

变量是和数值相关的名字，变量的主要作用是存取数据，提供存储信息的容器。可以调用变量的名称查看值或者改变值。因为 JavaScript 是无类型的，所以 JavaScript 的变量可以存放任何类型的值。

1．声明变量

在 JavaScript 程序中，使用变量之前需要声明。变量使用关键字 var 声明。也可以使用一个 var 关键字声明多个变量。例如：

```
var myname;      //声明一个变量名为 myname
var x,y;         //声明多个变量
```

在 JavaScript 中，变量也可以不声明，在使用时根据数据类型来确定变量的类型。例如：

```
myname="Sum";    //声明一个字符串变量
sum=100;         //声明一个整数变量
```

JavaScript 中变量名的命名规则如下。

- 第一个字符必须是一个 ASCII 字母（大小写均可），或一个下划线（_）。注意第一个字符不能是数字。
- 后续的字符必须是字母、数字或下划线。
- 变量名称一定不能是保留字。



JavaScript 区分大小写，例如一个名称为 myname 的变量和一个名称为 Myname 的变量是不同的。

2．变量的作用域

变量的作用域是在程序中定义变量的使用范围。全局变量的作用域具有全局性，即在 JavaScript 代码中，每个函数都可以使用该变量，有效范围从指定开始直到关闭页面才会结束。而在函数之内声明的变量，就只能在函数内部使用，称为局部变量。函数的参数也是局部变量，只能在函数体内使用。下面通过一个例子说明全局变量和局部变量的使用，在该例子中声明一个全局变量，然后在 checkscope()函数中声明一个同名的局部变量，当调用该函数时，则输出局部变量的值。具体实现如代码 11.1 所示。

代码 11.1　变量的作用域

```
<html>
<head>
<title>变量的使用</title>
</head>
<body>
<script type="text/javascript">
var scope ="mytest";  //声明一个全局变量
function checkscope()
{
  var scope="test";  //声明一个同名的局部变量
  document.write("输出局部变量的值: ",scope);  //使用的是局部变量，而不是全局变量
```

```
}
checkscope();
document.writeln("<br/>");
document.write("输出全局变量的值: ",scope);  //使用全局变量
</script>
</body>
</html>
```

将上述代码保存为 scope.html，打开该页面，可以看到调用 checkscope()函数输出局部变量的值。页面运行效果如图 11-1 所示。

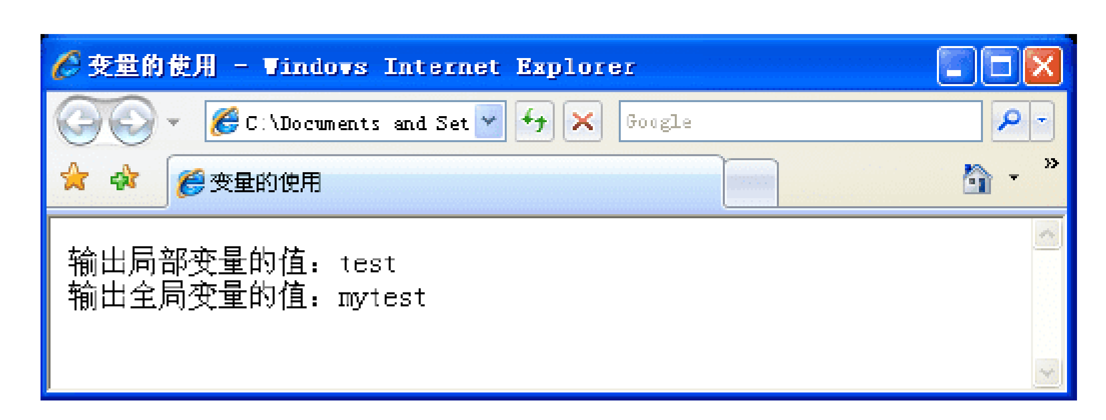
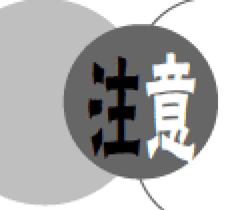


图 11-1　变量的使用



在声明全局变量时，可以不使用 var 关键字，但是在声明局部变量时，必须使用 var 关键字。

11.2.2　运算符

在 JavaScript 程序中，任何一个可以返回值的语句，都可以看作表达式。也就是说，表达式是一个短语，能够执行一个动作，并具有返回值。一个表达式通常由两部分构成，一部分是操作数，一部分是运算符。操作数就是在进行表达式计算时需要使用的数值。运算符就是表达式需要执行操作的类型。常见的运算符有算术运算符、赋值运算符、比较运算符和逻辑运算符等。

1. 算术运算符

算术运算符是最简单、最常用的运算符，用于进行通用的数学计算，如表 11-1 所示。

表 11-1　算术运算符

运算符	说明	示例	结果
+	加	i=2; j=2; i+j	4
–	减	i=5; j=2; i–j	3
*	乘	i=5; j=4; i*j	20
/	除	i=5; j=2; i/j	2.5
%	余数	i=5; j=2; i%j	1
++	递增	i=5; ++	6
– –	递减	i=5; – –	4

2．赋值运算符

赋值运算符用于更新变量的值，有些赋值运算符可以和其他运算符组合使用，对变量包含的值进行计算，然后用计算出的值更新变量。表 11-2 列出了这些赋值运算符。

表 11-2　赋值运算符

运算符	表达式	等价于
=	x=j	x=j
+=	x+=j	x=x+j
–=	x–=j	x=x–j
*=	x*=j	x=x*j
/=	x/=j	x=x/j
%=	x%=j	x=x%j

3．逻辑运算符

逻辑运算符通常用于执行布尔运算，常和比较运算符一起使用来表示复杂的比较运算，这些运算涉及的变量通常不止一个，而且常用于 if、while 和 for 语句中，如表 11-3 所示。

表 11-3　逻辑运算符

运算符	说明	示例
&&	与	4>2&&4<5 return true
		4>2&&4>5 return false
||	或	4>2||4>5　return true
		4>5||4>7　return false
!	非	!(4>2)　return false
		!(4>5)　return true

4．比较运算符

比较运算符用于对两个表达式进行比较，然后返回 boolean 类型的值。例如比较两个值是否相等或比较数字值的大小等，如表 11-4 所示。

表 11-4　比较运算符

运算符	说明	示例
==	等于	5==6 return false
!=	不等于	5!=8 return true
>	大于	5>8　return false
>=	大于等于	5>=6 return false
<	小于	4<6　return true
<=	小于等于	4<=6 return true

5．条件运算符

在 JavaScript 中，也支持 Java、C 和 C++中的条件表达式运算符“? :”，这个运算符是三

元运算符，它有 3 个部分，一个计算值的条件和两个根据条件返回的值。语法如下所示：

```
myname = (condition) ? value1 : value2;
```

例子：

```
myname =(name=="Anne")? "Dear Anne" :"Dear";
```

如果变量 name 的值等于 Anne，那么 myname 的值就为 Dear Anne。如果不为 Anne，那么 myname 的值就为 Dear。

6. 字符串运算符

字符串运算符在文字中使用得比较多。JavaScript 支持使用字符串运算符“+”对两个或多个字符串进行连接操作。下面举例说明：

```
var str1 = "What a very";
var str2 = "Good Nice!";
var str3 = str1+str2;       //结果为 What a veryGood Nice!
var str4 =str1+" "+str2;    //结果为 What a veryGood Nice!
```

11.2.3 数据类型

JavaScript 允许使用 3 种基本数据类型：数值、字符串和布尔值。还支持两种数据类型 null 和 undefined，它们各自只定义一个值。除这些基本的数据类型外，JavaScript 还支持复合数据类型：对象和数组。在 JavaScript 中的对象和数组虽然是同一种数据类型，但是行为却不相同。JavaScript 中的数据类型如表 11-5 所示。

表 11-5 JavaScript 中的数据类型

数据类型	数据类型名称	示例
number	数值类型	2、45、–123
string	字符串类型	'sum'、'Anne'
object	对象类型	Date、Document
boolean	布尔类型	true、false
null	空类型	null
undefined	未定义类型	today、tem

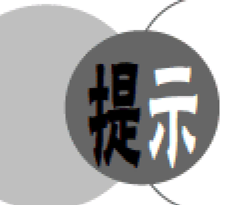
1. 数值

数值（number）是最基本的数据类型。JavaScript 数值类型表示一个数字，例如 5，12，–5，3e5 等。大于 0 的数称为正数，可以在正数前面加上正号（+），也可以省略。小于 0 的数称为负数，在前面加上负号（–），例如–16。在 JavaScript 中也可以包含指数，如 3e5，表示 3 乘以 10 的 5 次方。同时还可以以十进制、八进制和十六进制作为基数来表示。

- 八进制正数使用一个前缀 0 指明，并可以包含从 0-7 的数字。如果该数字包含数值 8 或 9，那么该数字是一个十进制数。由于某些 JavaScript 实现支持八进制数，而有些则

不支持，所以最好不要使用以 0 开头的整数，因为不知道 JavaScript 实现是将其解释为十进制，还是解释为八进制。

- 十六进制的数字以 0X 或者 0x 开头，并且包含 0-9 之间的任何数字，也可以包含 a（A）到 f（F）的任何字母，用来表示 0-15 之间的某个值。



JavaScript 只有一种数字类型，而且内部使用的是 64 位浮点型，等同于 C#或 Java 中的 double。

2. 字符串

字符串是使用 Unicode 字符、数字和标点符号等组成的序列，用于表示文本。一个字符串也是 JavaScript 中的一个对象，有专门的属性。下面是字符串的示例：

```
var str='http://www.itzcn.com';
var str1="123123";
```

字符串变量的值必须得写在一行中，如果将其放在两行中，可能会将其截断。如果必须在字符串中添加一个换行符，可以使用字符序列“\n”。在 JavaScript 的字符串中，反斜线（\）具有特殊的用途。在反斜线后面加一个字符就可以表示在字符串中无法出现的字符。JavaScript 中的转义字符如表 11-6 所示。

表 11-6　JavaScript 中的转义字符

转义字符	含义	转义字符	含义
\b	后退一格	\t	制表
\f	换页	\'	单引号
\n	换行	\"	双引号
\r	返回	\\	反斜线

3. 布尔值

布尔数据类型表示一个逻辑数值，可取值为 true 或 false。布尔值通常在 JavaScript 程序中表示比较所得的结果。例如：

```
a==5;
```

这行代码测试变量 a 的值是否和 5 相等，如果相等，比较的结果就是 true，否则结果就是 false。布尔值通常用于 JavaScript 的控制结构中。例如 JavaScript 的 if esle 语句就是在布尔值为 true 时执行一个动作，为 false 时执行另一个动作。

11.3　流程控制语句

程序的执行流程通常是线性的，即按顺序逐条处理每个语句。为了满足程序的逻辑需求，

程序的执行流程经常需要改变，因此任何一种语言都提供对程序执行流程的控制。JavaScript控制语句包括条件语句和循环语句。

11.3.1 条件语句

使用条件语句可以根据表达式的值有条件地执行一组语句。JavaScript 提供多种条件语句，执行原理相同，仅在形式上有所差异。条件语句包括 if else 语句和 switch 语句。

1. If else 语句

if else 语句是基本的条件语句，可以使 JavaScript 有选择地执行语句。if 语句执行过程如图 11-2 所示。

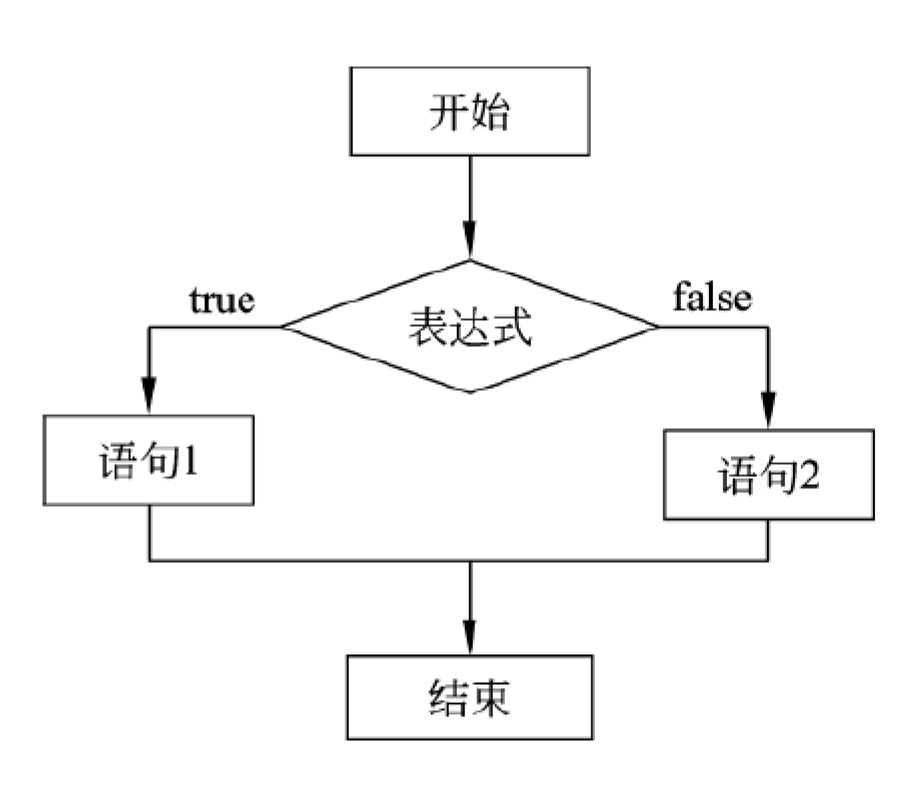


图 11-2 if else 语句流程图

在 if else 语句流程图中，根据表达式返回的结果，执行相应的语句。如果返回结果为 true，执行语句 1，如果返回结果为 false，执行语句 2。下面通过示例说明如何在 JavaScript 中使用 if else 语句。打开一个记事本，输入代码 11.2 所示内容，然后将文件保存为 HTML 文件。

代码 11.2 if else 语句

```
<html>
<head>
<title>if else 语句的使用</title>
</head>
<body>
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if(time<10)
{
  document.write("<b>早上好! </b>");
}
else
{
  document.write("今天是美好的一天! ");
}
</script>
</body>
</html>
```

在上述代码中，首先实例化一个 Date 对象，然后使用该对象调用 getHours()方法，将获取的时间赋给变量 time，接着使用 if 语句判断当前时间是否小于 10，如果返回结果为真，那

么在页面上输出“早上好！”，否则，输出“今天是美好的一天！”。页面运行效果如图 11-3 所示。

2. switch 语句

switch 是多分支语句，用于判断条件表达式有多种输出结果的情况。在执行 switch 语句时将处理表达式，并根据表达式的结果选择执行相应的一个或多个语句。switch 语句执行过程如图 11-4 所示。

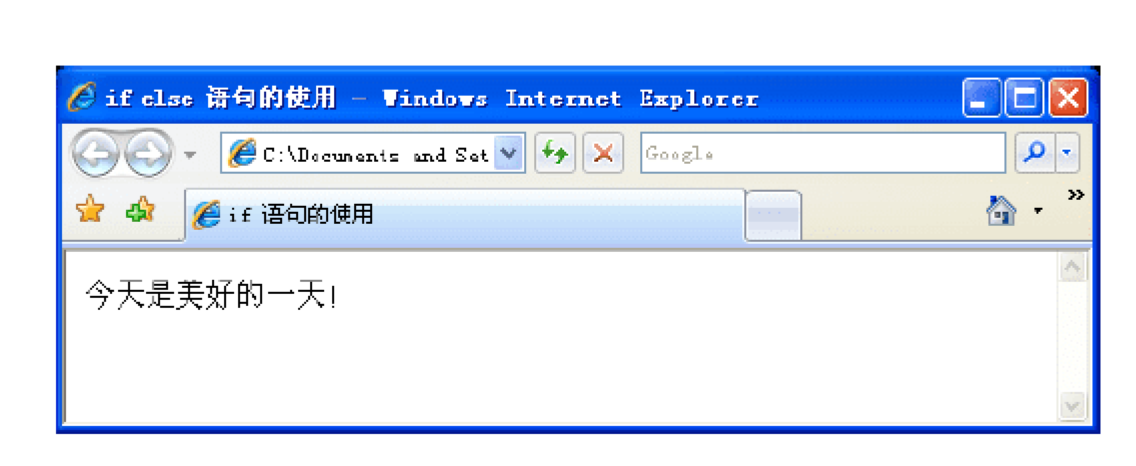


图 11-3　if else 语句

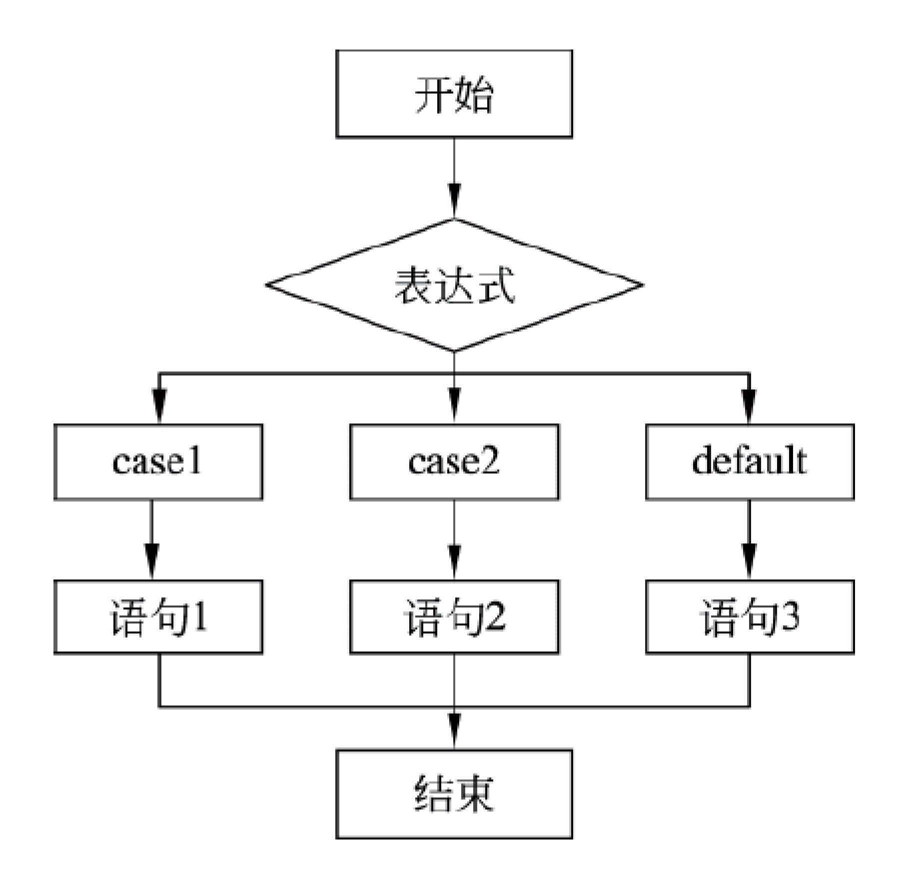


图 11-4　switch 语句流程图

在 switch 语句流程图中，根据 switch 表达式返回的值，查找与之相对应的 case 语句并执行。如果没有找到与表达式相匹配的 case 语句，那么执行 default 后的语句。下面通过示例说明如何在 JavaScript 中使用 switch 语句。打开一个记事本，输入代码 11.3 所示内容，然后将文件保存为 HTML 文件。

代码 11.3　switch 语句

```
<html>
<head>
<title>switch 语句的使用</title>
</head>
<body>
<script type="text/javascript">
var d=new Date();
theDay=d.getDay();
switch (theDay)
   {
   case 5:
     document.write("今天是周五，你打算做什么呢？");
     break;
   case 6:
     document.write("哈哈，今天周六，可以爬山了。");
     break;
```

```
      case 0:
        document.write("周日，我在家帮妈妈做家务。");
        break;
      default:
        document.write("我这个星期都干了些什么？");
    }
  </script>
  </body>
  </html>
```

在上述代码中，实例化一个 Date 对象，然后使用该对象调用 getDay()方法，并将获取的日期赋给变量 day，然后根据表达式的返回值，输出不同的问候代码。页面运行效果如图 11-5 所示。

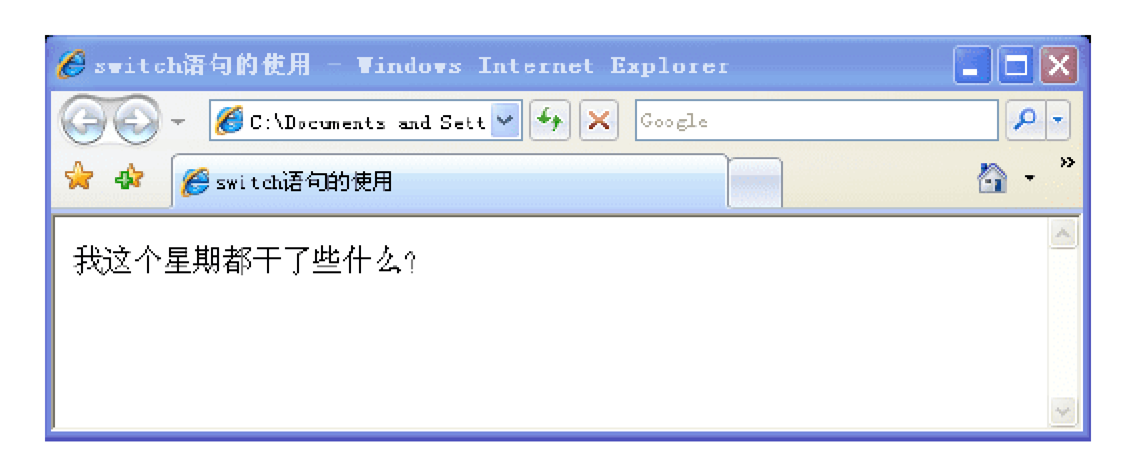


图 11-5 switch 语句的使用

11.3.2 循环语句

在编写程序的过程中，有时需要重复执行某些动作，这时需要用到循环语句。JavaScript 中的循环结构有以下几种：while 语句、do while 语句、for 语句和 for in 语句。

1. while 语句

如果不确定循环体执行的次数，而是由一些复杂的规则控制，则应该使用 while 语句。while 语句执行过程如图 11-6 所示。

在 while 语句流程图中，根据表达式返回的结果，执行相应的语句。如果返回值为 true，执行语句 1，当循环体执行完毕，则完成一次循环，然后重新判断表达式是否为真，一直重复这个过程，直到表达式返回的值为 false 时，退出 while 循环。

开始
语句1
true
表达式
false
结束

图 11-6 while 语句流程图

2. do while 语句

do while 语句与 while 语句相似，do while 语句是在循环底部检查循环表达式，而不是在循环顶部进行检测，这就意味着循环体至少会被执行一次。do while 语句执行过程如图 11-7 所示。

3. for 语句

for 语句提供一个循环结构，这个结构通常比 while 语句方便。for 语句指定一个表达式 1、

条件表达式 2 和表达式 3，每次循环之前都要测试条件表达式返回结果是否为 true，如果为 true，则执行循环内的语句，如果为 false，则不执行循环内的语句，而是执行紧跟在循环后的第一行语句。当执行循环时，计算器变量在下一次重复循环前被更新。for 语句执行过程如图 11-8 所示。

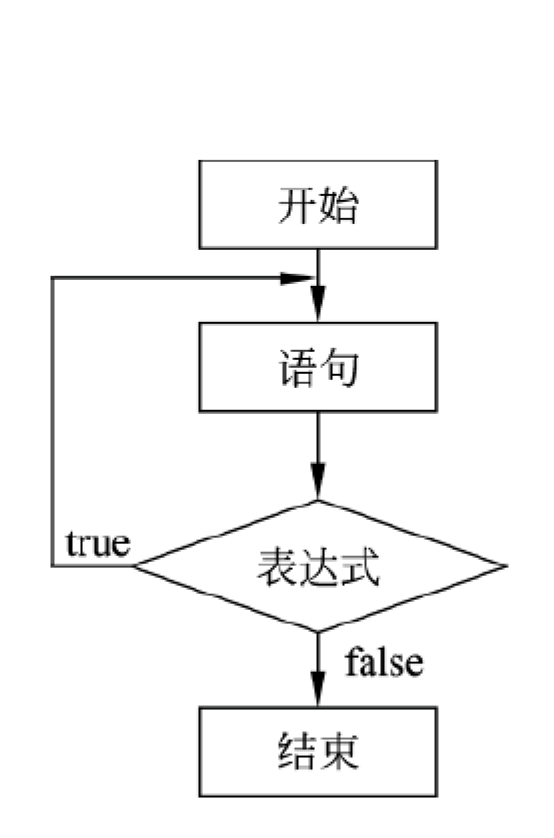


图 11-7　do while 语句流程图

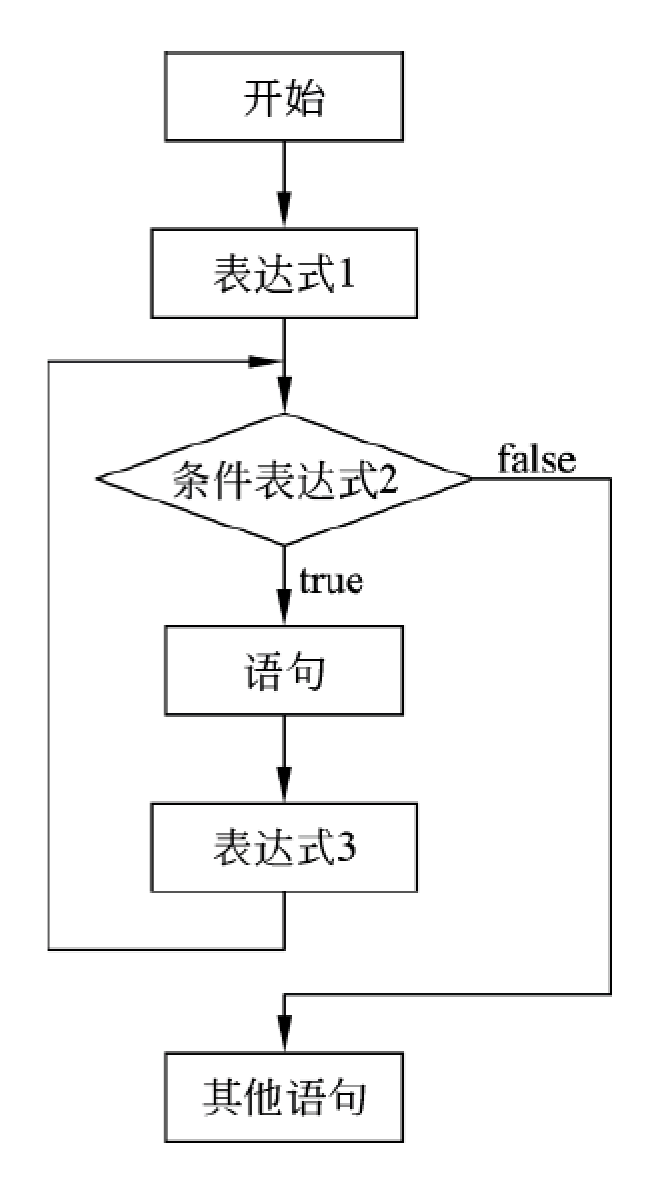


图 11-8　for 语句流程图

下面通过一个例子说明 for 循环的使用方法，输出 0～10 之间的整数。具体实现如代码 11.4 所示。

代码 11.4　for 语句

```
<html>
<head>
<title>for 语句的使用</title>
</head>
<body>
<script type="text/javascript">
for(var i=0;i<=10;i++)
{
  document.write(i);
}
</script>
</body>
</html>
```

4. for in 语句

在 JavaScript 中关键字 for 有两种使用方法。一种是上面介绍的 for 语句，另一种是 for in

语句。该循环语句可以对数据或对象的所有属性进行循环操作，语法格式如下所示：

```
for(变量 in 对象)
{
 语句块;
}
```

下面通过示例讲解 for in 语句的使用，该例子用于输出定义对象的各个属性值。具体实现如代码 11.5 所示。

代码 11.5　for in 语句

```
<html>
<head>
<title>for in 循环</title>
</head>
<body>
<script type="text/javascript">
//创建一个对象，以及 4 个属性
var mybook = new Object();
mybook.name="红楼梦";
mybook.author="曹雪芹";
mybook.price=35;
mybook.ISBN="013457";
//遍历对象中的所有属性
for (book in mybook)
{
  document.write(book +"="+mybook[book]);
  document.write("<br/>");
}
</script>
</body>
</html>
```

在上述代码中，实例化一个 mybook 对象，然后为该对象添加 4 个属性，接着使用 for in 语句遍历 mybook 对象中的属性，将代码保存为“for in 语句.html”，页面运行效果如图 11-9 所示。

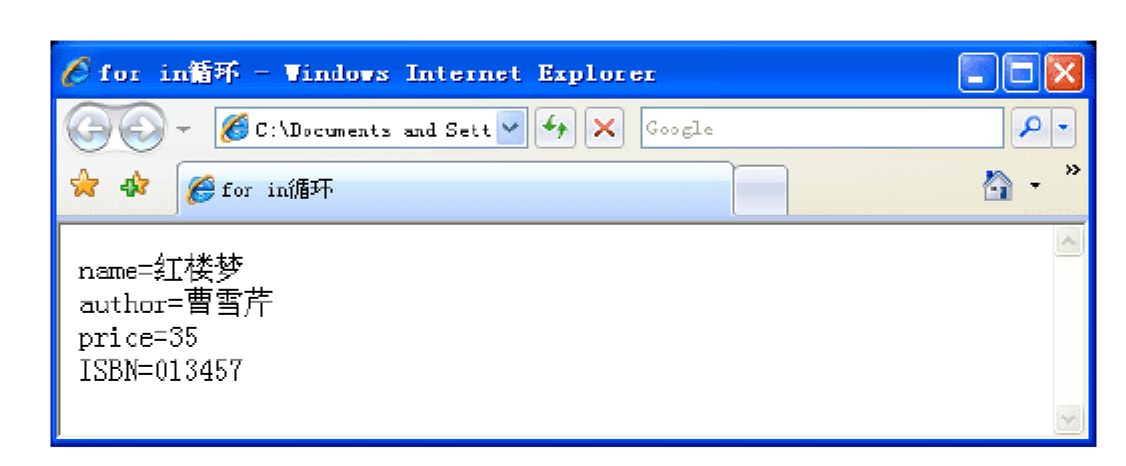


图 11-9　for in 语句的使用

11.3.3　其他语句

在 JavaScript 中，使用 break 语句可以无条件地从当前循环结构或者 switch 结构的语句块中中断程序并退出当前循环。break 语句后面可以跟一个标签名，可以跳转到这个标签名的语句尾部或终止这个语句。

continue 语句的工作方式与 break 语句有点类似，都可以结束当前循环，也可以得到语句标记。与 break 不同的是 continue 语句没有使应用程序退出循环，而是结束当前循环并跳转到循环开始的地方继续执行程序。

return 语句用于向函数返回一个值，如果这个值是函数的表达式，那么将先执行表达式的运算，再将结果返回。return 语句可以用在函数体中。



continue 语句只能用在 while 语句、for 语句或者 for in 语句的循环体内，在这之外使用将会引起语法错误。

11.3.4　异常处理

异常处理是一个强大的、多用途的错误处理和恢复系统，JavaScript 中使用 try catch finally 语句处理异常。一个 try 语句包含一组语句块，在这组语句块中可能会发生异常，catch 语句定义如何处理错误，finally 语句中包含始终被执行的代码。

一般来说，代码要执行一组语句块，如果执行失败，就会跳转到 catch 语句块。如果没有错误发生，就跳过 catch 语句。finally 语句在 try 和 catch 语句执行完毕之后执行。try catch finally 语句的语法格式如下：

```
try {
// 语句块;
}
catch{
// 语句块;
}
finally{
// 语句块;
}
```

这个结构中 catch 和 finally 块是可选的，但是如果没有 catch 块，异常处理语句将没有意义。下面是一个使用异常处理语句的示例，如代码 11.6 所示。

代码 11.6　异常处理

```
<html><head>
<title>异常处理</title>
```

```
<script type="text/javascript">
function myCatch() {
try {
if (document.forms[0].firstName.value == '') {
}
}
catch(e) {
document.write("发生了一个错误.<br/>");
document.write("请联系管理员.<br/>");
document.write("错误消息描述: "+e.message);
}
finally {
document.forms[0].submit();
}
return 0; }
</script></head>
<body>
<form id="frmTest">
Name: <input id="fullName" name="fullName" type="text"><br/>
Address: <input id="contactNumber" name="contactNumber" type="text"><br/>
<input type="button" value="Submit" onclick="myCatch();">
</form>
</body>
</html>
```

在上述代码中使用 try catch finally 语句，由于在页面中没有定义 firstName，所以会发生一个异常。保存文件之后，在浏览器中查看运行效果。如图 11-10 所示。

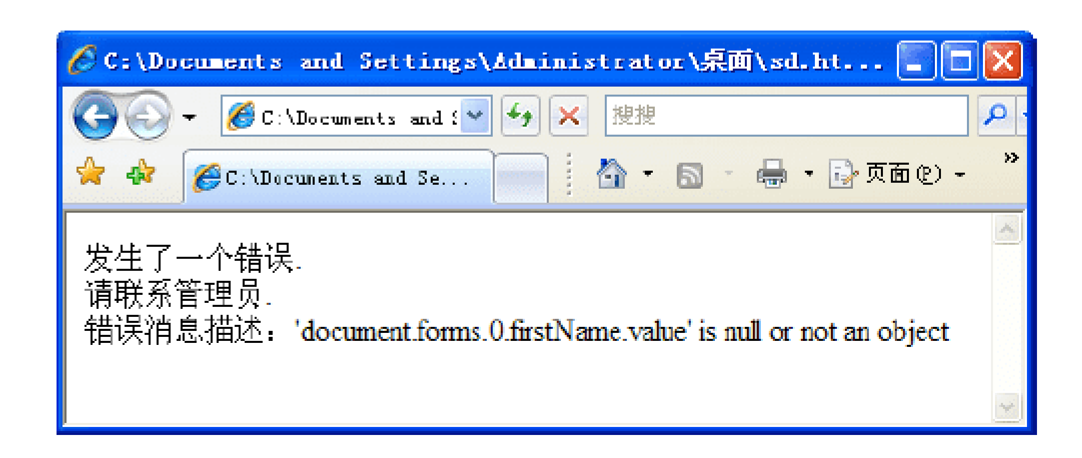


图 11-10 使用异常处理

11.4 函数

函数对于任何一种计算机语言来说都是非常重要的。在 JavaScript 语言中，函数是一个既重要又复杂的部分。JavaScript 函数可以封装那些在程序中可能要多次用到的模块，并可作为事件驱动的结果而调用程序，从而实现一个函数与相应的事件驱动相关联。

11.4.1　定义和调用函数

定义函数最常用的方法就是调用 function 语句。该语句由关键字 function 构成。定义 JavaScript 函数的语法格式如下：

```
function 函数名(参数)
{
  函数体;
  return 值;
}
```

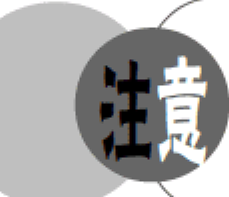
下面通过示例演示如何定义函数。在该例子中定义两个函数 Message()和 Sum()，由于在 Message()函数中没有 return 语句，所以没有返回值，在 Sum()函数中，使用 return 语句返回 3 个数相加的和，具体实现如代码 11.7 所示。

代码 11.7　定义函数

```
<html>
<head>
<title>定义函数</title>
</head>
<body>
<script type="text/javascript">
//由于该函数没有 return 语句，所以没有返回值
function Message(msg)
{
  document.write(msg,'<br/>');
}
//该函数是计算 3 个数的和
function Sum(a,b,c)
{
  return a+b+c;
}
Message("Hello World");
Message("3 个数的和是: "+Sum(1,2,3));
</script>
</body>
</html>
```

函数定义好以后，可以直接调用。在上述代码中，分别为 Message()和 Sum()函数传递参数，然后将代码保存为“调用函数.html”。打开文件，可以在页面中看到两条输出语句，如下所示：

```
Hello World
3 个数的和是: 6
```



参数变量只有在执行函数时才会被定义，如果函数返回，那么它们就不再存在。

11.4.2　基于对象的函数

JavaScript 中的函数不仅可以在页面载入时静态创建，也可以在函数被调用时创建，这说明函数具有对象的特征，这样的函数称为函数对象。下面代码创建一个函数对象：

```
var aaa = new Function("a","b","c","return a+b+c");
document.write(aaa(1,2,3));
```

当运行这段代码之后，将创建一个新的函数。这个函数的使用和定义与普通的函数定义语法类似。new 和 Function 是必须的。Function 构造函数中可以接收任意多个字符串类型的函数参数。最后一个参数是函数的主体，决定函数要完成的功能。其中可以包含多个 JavaScript 语句，语句之间用分号隔开。

基于对象的函数的另一种表示方法如下所示：

```
var bbb = function(a,b,c){return a+b+c;};
```

下面通过示例演示如何使用 function 语句和 Function 构造函数创建函数，以及调用方法。首先打开一个记事本，然后输入如代码 11.8 所示代码。最后将文件保存为“创建和调用函数.html”。

代码 11.8　创建和调用函数

```
<html>
<head>
<title>创建和调用函数</title>
<script language="JavaScript">
function sqrt(x)
{
  return x*x;
}
var sqrt1 = new Function("x","x","return x*x");
var sqrt2 = function(x,x){return x*x;};
</script>
</head>
<body>
<h2 style="text-align:center">创建和调用函数</h2>
<h3>调用 sqrt 函数</h3>
<script>
document.writeln("函数 sqrt(3)计算结果是: ",sqrt(3));
</script>
<h3>调用 sqrt1 函数</h3>
```

```
<script>
document.writeln("函数 sqrt1(3)计算结果是: ",sqrt1(3,3));
</script>
<h3>调用 sqrt2 函数</h3>
<script>
document.writeln("函数 sqrt2(3)计算结果是: ",sqrt2(3,3));
</script>
</body>
</html>
```

上述代码中，创建了 3 个函数，然后在 HTML 页面中添加调用语句，并将结果输出。在浏览器中查看结果，如图 11-11 所示。



图 11-11　创建和调用函数

11.4.3　系统函数

JavaScript 中的系统函数又称为内部方法或内置函数。内置函数属于任何对象，在 JavaScript 语句的任何地方都可以使用这些函数。系统函数不需要创建，也就是说，用户可以在任何需要的地方调用它，如果函数有参数，还需要在括号中指定传递的值。下面通过示例演示如何使用 JavaScript 中的一些系统函数，以及如何调用系统函数。

1. eval()函数

eval()函数用于接收一个字符串形式的表达式，并返回表达式的值。下面通过示例演示 eval()函数的使用，在该示例中，定义一个字符串表达式，然后使用 eval()返回表达式的值。具体实现如代码 11.9 所示。

代码 11.9　eval()函数

```
<html>
<head>
<title>eval()函数的使用</title>
<script language="JavaScript">
```

```
var myString="5+11";
var mySum =eval(myString);
document.write(myString+"="+mySum);
</script>
</head>
</html>
```


将上述代码保存为“eval()函数.html”，然后运行程序，在浏览器中输出的结果如下所示：

```
5+11=16
```

2．parseFloat()和 parseInt()函数

parseFloat()函数从一个字符串中获取一个浮点值，如果在字符串中不存在除了数字、符号、小数点和指数以外的字符，parseFloat()函数就停止转换，返回已有结果。parseInt()返回不同的进制数，默认是十进制。下面通过示例讲解这两个函数的使用，具体实现如代码 11.10 所示。

代码 11.10　parseFloat()和 parseInt()函数

```
<html>
<head>
<title>parseFloat 和 parseInt 函数的使用</title>
<script type="text/javascript">
//parseFloat()函数
var floatStr="5.16";
document.write("使用 parseFloat 函数: "+"<br/>")
document.write((floatStr+1)+"<br/>");
document.write((parseFloat(floatStr)+1)+"<br/>");
//parseInt()函数
document.write("使用 parseInt()函数: "+"<br/>");
document.write("默认情况下 15 表示:"+parseInt('15')+"<br/>");
document.write("8 进制下 15 表示: "+parseInt('15',8)+"<br/>");
document.write("16 进制下 15 表示: "+parseInt('15',16)+"<br/>");
//输出一个数在不同进制下表示的值
document.write("二进制:"+parseInt('00110011',2)+"<br/>");
document.write("八进制:"+parseInt('00110011',8)+"<br/>");
document.write("十六进制:"+parseInt('00110011',16)+"<br/>");
document.write("十进制:"+parseInt('00110011',10));
</script>
</head>
</html>
```

上述代码中分别使用 parseFloat()和 parseInt()函数对数值进行转换。保存文件之后，运行程序，在浏览器中输出的结果如图 11-12 所示。

3．isNaN()函数

当 JavaScript 使用 parseInt()函数和 parseFloat()函数遇到一个不能转换为数字的字符串时，

将自动返回一个叫 NaN 的结果。isNaN()函数测试这两个函数的返回结果是否为 NaN，如果是，函数返回 true。下面通过示例演示该函数的使用，如代码 11.11 所示。

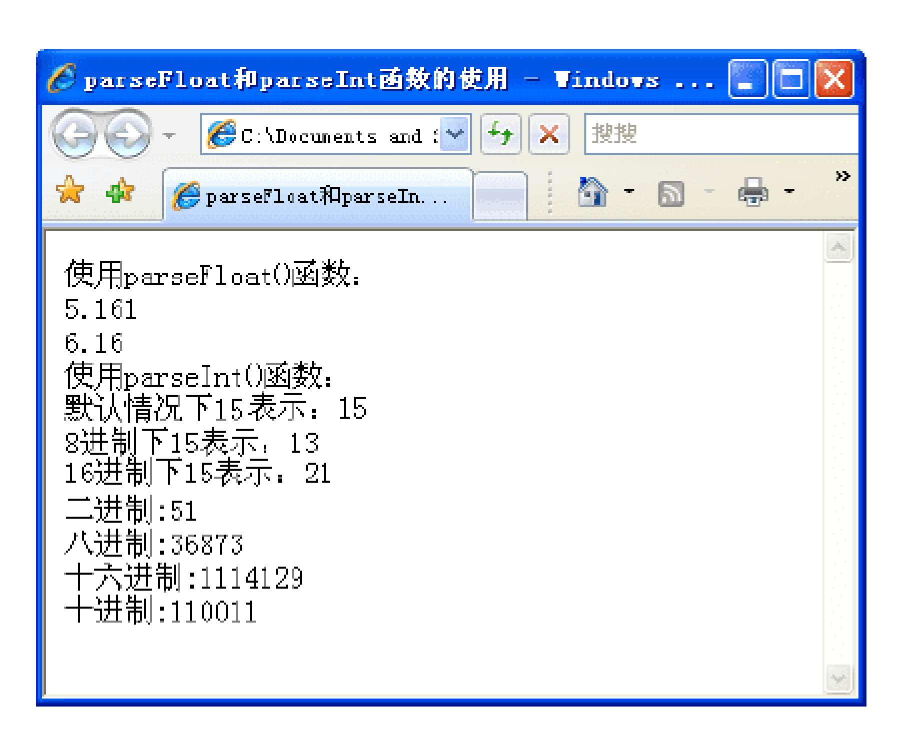


图 11-12　运行结果

代码 11.11　isNaN()函数

```
<html>
<head>
<title>isNaN()函数的使用</title>
<script type="text/javascript">
var LabelName=parseInt('admin518');
if(isNaN(LabelName))
{
  //如果 LabelName 不是数值，执行下面语句
    document.write("parseInt('admin518')的结果是: "+LabelName);
}
else
{
    document.write("parseInt('admin518')的结果是数值");
}
</script>
</head>
</html>
```

在上述代码中，将 parseInt()函数转换字符串的结果保存在 LabelName 变量中，然后使用 isNaN()函数判断 LabelName 是否为数值。保存文件，运行程序，结果如下所示：

```
parseInt('admin518')的结果是: NaN
```

4．escape()和 unescape()函数

escape()函数返回对一个字符串进行编码后的结果字符串。unescape()函数将一个用 escape()函数编码的结果字符串解码为原始字符串并返回。下面通过示例演示这两个函数的使用，如代码 11.12 所示。

代码 11.12 escape()和 unescape()函数

```
<html>
<head>
<title>escape()和 unescape()函数的使用</title>
<script type="text/javascript">
var text = escape("Lee Anne");
document.write("Lee Anne 经 escape 方法处理后的结果为: "+text);
document.write("<br/>");
var unescapeStr =unescape(text);
document.write("unescapeStr:"+unescapeStr);
</script>
</head>
</html>
```

在上述代码中，使用 escape()函数对字符串进行编码，然后再使用 unescape()函数解码，并返回原始字符串。保存文件，运行程序，结果如下所示：

```
Lee Anne 经 escape 方法处理后的结果为: Lee%20Anne
unescapeStr:Lee Anne
```

在 JavaScript 中还有一些其他系统函数，这些函数在使用时不需要创建任何实例，可直接调用。

11.5 事件机制

事件是浏览器响应用户交互操作的一种机制，JavaScript 的事件处理机制可以改变浏览器响应用户操作的方式，这样就可以开发出具有交互性、响应性和易于使用的网页。浏览器为响应某个事件而进行的处理过程，叫做事件处理。

11.5.1 事件概述

在程序设计中，当文档或其中某些元素发生某些事件时，Web 浏览器就会生成一个事件（event）。例如在浏览器装载完一个文档，在用户将鼠标移到一个超链接或者单击 Submit 按钮时，浏览器都会生成事件。如果 JavaScript 应用程序注重特定文档元素的特定类型的事件，那么就会为这个元素的类型事件注册一个事件处理程序（event handler），即一个 JavaScript 函数或代码段。然后在发生特定事件时，浏览器将调用处理程序代码。目前使用的有 4 种完全不同的、不兼容的事件处理模型，如下所示。

□ **原始事件模型**

这是一种简单的事件处理模型。HTML 4 标准已部分地将其收录到标准中，通常看作 0 级 DOM API 的一部分。尽管其特性有限，但是所有启用 JavaScript 的浏览器都支持，因此具

有可移植性。

- **标准事件模型**

标准事件模型是一种强大的、具有完整特性的事件模型，2 级 DOM 标准对其进行标准化。

- **Internet Explore 事件模型**

该事件模型由 Internet Explore 4 及其后续版本实现，具有标准事件模型的许多高级特性，但不具备全部特性。Microsoft 公司参与了 2 级 DOM 事件的创建，而且有足够的事件在 IE 5 和 IE 6 中实现这一标准事件模型，但它仍然坚持使用自己特有的事件模型。这说明，使用高级事件处理特性的 JavaScript 开发者必须为 IE 浏览器编写特定的代码。

- **Netscape 4 事件模型**

该事件由 Netscape 4 实现，尽管标准事件模型已经取代 Netscape 4 事件模型，但是 Netscape 6 仍然支持。其具有标准事件模型的某些高级特性，但不是全部特性。适合既要具有高级事件处理特性，又要与 Netscape 4 兼容的 JavaScript 开发者。

11.5.2　事件处理程序

事件定义用户与页面交互时产生的各种操作，例如单击按钮时，就产生一个单击（click）操作事件。浏览器在程序运行时，大部分时间都在等待交互事件的发生，并在事件发生时自动调用事件处理函数，完成事件处理过程。

事件不仅可以在用户交互过程中产生，而且浏览器本身的一些动作也可以产生事件，例如载入一个页面时，就会发生 load 事件，卸载一个页面时，就会发生 unload 事件等。

事件的处理程序可以是任意 JavaScript 语句，但是一般用特定的自定义函数（function）来处理事件。其基本格式与函数完全一样，可以将前面介绍的所有函数作为事件处理程序，格式如下：

```
function 事件处理名(参数名)
{
  事件处理语句集;
}
```

事件处理是对象化编程的一个重要环节，没有事件处理，程序就不会多样化，缺乏灵活性。事件处理的过程可以这样表示：

```
发生事件 – 启动事件处理程序 – 事件处理程序做出反应
```

但是调用处理程序之前，必须提前说明什么事件调用什么处理程序，否则这个流程就不能进行下去。指定事件调用程序可以通过 3 种方法。

1. 在 HTML 标签中直接指定

在 HTML 标签中直接指定调用的事件，是最简单，也是使用最普遍的一种方法，使用格式如下：

```
<html    触发的事件="调用的处理程序">
```

例如：

```
<body onload="JavaScript:alert('欢迎您的到来！')" onunload="JavaScript:alert('
欢迎下次光临')">
```


在这个 body 标签事件中，当打开文档时，会弹出一个对话窗口“欢迎您的到来”，当关闭文档时，则会弹出一个对话窗口“欢迎下次光临”。

2. 绑定对象的特定事件

绑定对象的特定事件，这种方法一般很少使用，但是在某些场合还是很有用的，使用格式如下：

```
<script language="JavaScript" for="对象" event="事件">
//程序代码
</script>
```

例如：

```
<script language="JavaScript" for="window" event="onunload">
alert('欢迎下次光临');
</script>
```

在这段代码中绑定一个将窗口关闭的事件。当关闭页面时，会弹出一个对话框“欢迎下次光临”，当用户单击确定之后，才会将这个页面关闭。

3. 在 JavaScript 中说明

在 JavaScript 中说明，该方法的使用格式如下所示：

```
对象 . 事件 = 调用的方法
```

例如：

```
function Errors()
{
  return true;
}
window.onerror=Errors;
```

该示例将 Errors()函数定义为 window 对象的 onerror 事件的处理程序，作用就是当 window 发生什么错误都会被忽略。

11.5.3 事件驱动

JavaScript 事件驱动中的事件通过鼠标或热键的动作引发。主要事件如表 11-7 所示，并列出了哪些 HTML 标记支持这些事件处理属性。

表 11-7　JavaScript 事件处理属性

事件名称	触发条件	支持标记
onabort	图像装载被中断	<img>
onblur	失去输入焦点时	<button>、<input>、<select>、<body>、<textarea>
onchange	选择<select>标记中的选项或其他表单标记失去焦点，并且由于它获得了焦点而使值发生改变时	<input>、<select>、<label>、<textarea>
onclick	单击鼠标并释放，发生在 mouseup 事件后，若返回 false，则可以取消默认动作	大多数标记
ondbclick	双击鼠标时	大多数标记
onerror	在装载图像的过程中发生错误时	<img>
onfocus	标记得到输入的焦点	<input>、<lable>、<body>、<button>、<select>、<textarea>
onkeydown	键盘键被按下时，若返回 false，则可以取消默认动作	表单元素、<body>
onkeypress	键盘键被释放时，若返回 false，则可以取消默认动作	表单元素、<body>
onkeyup	键盘键被释放时	表单元素、<body>
onload	文件加载完成时	<body>、<img>、<object>、<frameset>、<iframe>
onmousedown	单击鼠标时	大多数标记
onmousemove	鼠标移动时	大多数标记
onmouseout	鼠标离开时	大多数标记
onmouseover	鼠标移动到标记上时	大多数标记
onmouseup	释放鼠标左键时	大多数标记
onreset	表单请求被重置时，若返回 false 则阻止重置	<form>
onresize	调整窗口大小时	<body>、<frameset>
onselect	选择文本时	<input>、<textarea>
onsubmit	请求提交表单时，若返回 false 则阻止提交	<form>
onunload	关闭文件或框架时	<body>、<frameset>

可以将表中所列的这些事件分为两类，原始事件和语义事件。原始事件是在用户移动鼠标、单击鼠标或者按下键盘时触发。这些事件只描述用户的动作，而没有其他含义。语义事件的含义比较复杂，通常只有在特定的环境中才会被触发。例如在浏览器加载完文件或单击提交表单时，语义事件通常作为事件的附加属性触发。

11.5.4　事件处理的使用方法

JavaScript 事件像 HTML 标签的属性一样经常使用。事件是发生在 HTML 元素上的某些行为，例如什么时候被点击或者什么时候失去焦点。下面通过示例演示如何使用 JavaScript 的一些事件。

1．onblur 和 onchange 事件

onblur 和 onchange 事件都是当标记失去输入焦点时触发，但 onchange 是失去焦点后，如果 value 发生改变则触发。下面通过示例演示 onblur 和 onchange 事件的使用，具体实现如代

码 11.13 所示。

代码 11.13　onblur 和 onchange 事件

```
<html>
<head>
<title>onblur 和 onchange 事件</title>
</head>
<body>
<input type="text" value="输入内容" onblur="alert('onblur')" onchange="alert
('onchange')"/>
</body>
</html>
```

保存文件后，在浏览器中查看运行结果。只要文本框失去焦点就会触发 onblur 事件。当文本框进行修改后，再次失去焦点就会触发 onchange 事件。

2．onclick 事件

鼠标单击事件是最常见的事件之一，当用户在某些表单元素上单击鼠标按钮时，便会触发 onclick 事件。下面通过示例演示 onclick 事件的使用，具体实现如代码 11.14 所示。

代码 11.14　onclick 事件

```
<html>
<head>
<title> onclick 事件</title>
</head>
<body>
<input type="text" id="textbox1" /><input type="button" value="提交" onclick=
"if (document.getElementById
('textbox1').value=='stand') location='http://www.itzcn.com'; else alert('
错误消息');" />
</body>
</html>
```

在上述代码中，判断文本框中输入的值，如果输入的值为 stand，那么页面跳转到 http://www.itzcn.com，否则，弹出一个对话框提示信息为“错误消息”。

3．onload 和 onunload 事件

onload 事件是全局对象 window 的一个成员，允许指定一个 JavaScript 函数，这个函数将在整个页面，包括 HTML 标签、图片以及脚本全部下载到浏览器之后被执行。和大多数事件不同，window.onload 事件的处理函数可以通过 JavaScript 显式地附加指定，或者也可以在 HTML 内容里显式指定。onunload 事件是关闭当前网页时发生的事件。下面通过示例演示 onload 和 onunload 事件的使用，具体实现如代码 11.15 所示。

代码 11.15　onload 和 onunload 事件

```
<html>
<head>
<title> onload 和 onunload 事件</title>
</head>
<body onLoad="JavaScript:alert('欢迎光临我的网站！');" onUnload="JavaScript:
alert('谢谢，再见！');">
</body>
</html>
```

保存文件，然后在浏览器中查看运行结果。当用户打开页面时，会弹出一个对话框，提示信息为“欢迎光临我的网站！”，当用户关闭页面时，则弹出一个对话框，提示信息为“谢谢，再见！”。这是因为分别触发了 onload 事件和 onunload 事件。

4．onselect 事件

onselect 事件当文本框的内容被选中时发生。下面通过示例演示该事件的使用，当用户试图选中文本框中的内容时，弹出一个提示信息对话框，具体实现如代码 11.16 所示。

代码 11.16　onselect 事件

```
<html>
    <head>
    <title>onselect 事件</title>
    </head>
    <body>
    <form name="test">
       <textarea name="textarea1" rows="5" cols="35" onselect="alert('文本框
       中的文本不能复制！')">
        使用 onselect 事件
       </textarea>
    </form>
    </body>
</html>
```

保存文件，在浏览器中查看运行效果。当用户试图选择文本框中的信息时，会弹出一个对话框提示“文本框中的文本不能复制！”。

5．onmouseover 和 onmouseout 事件

下面通过实例演示鼠标事件的使用。在该示例中，当鼠标移动到或离开指定标记时触发事件，具体如代码 11.17 所示。

代码 11.17　onmouseover 和 onmouseout 事件

```
<html>
    <head>
```

```
    <title>onmouseover 和 onmouseout 事件</title>
    </head>
    <body>
      <font onmouseover="alert('这是什么？');" >My Program!</font>
      <font onmouseout="alert('鼠标已经离开');">下午好!</font>
    </body>
</html>
```

保存文件后，在浏览器中查看运行效果。当鼠标放在“My Program!”上时，会弹出一个对话框提示信息为“这是什么？”。当鼠标离开“下午好!”之后，则弹出一个对话框提示信息为“鼠标已经离开”。

6. onsubmit 事件

用户提交表单时（通常使用【提交】按钮），就会触发表单的 onsubmit 事件。该事件在实际提交表单之前发生。因此，该事件的事件处理程序可以通过返回 false 来阻止表单的提交。onsubmit 事件可用于验证表单输入项的正确性，下面通过示例演示该事件的使用，具体实现如代码 11.18 所示。

代码 11.18 onsubmit 事件

```
<html>
<head>
<title>onsubmit 事件</title>
    <script type="text/javascript">
        function check(){
            if(document.test.t.value == ""){
                alert("空值");
                return false;
            }
            return true;
        }
    </script>
</head>
<body>
<form name="test" onsubmit="return check();">
    <input type="text" name="t">
<input type="submit" value="提交">
</form>
</body>
```

在上述代码中，单独定义一个 check()函数。如果在事件处理程序中有多个语句，那么就可以将这些语句定义在一个函数体中，然后在 HTML 事件处理属性中调用这个函数。保存文件，在浏览器中查看运行结果。在文本框中什么内容都没有输入的情况下，单击【提交】按钮则会弹出一个对话框，提示信息为“空值”。

11.5.5　使用 this 关键字

在面向对象编程语言中，对于 this 关键字都非常熟悉，例如 C++、C#和 Java 等都提供这个关键字。虽然在开始学习时觉得比较难，但只要理解，使用起来就非常方便。在 JavaScript 中 this 关键字是比较重要的一个关键字，会随调用对象而发生改变，始终与当前对象的上下文保持一致。this 关键字通常在对象的构造函数中使用，用于引用对象。下面通过实例演示 this 关键字的使用，具体如代码 11.19 所示。

代码 11.19　this 关键字的使用

```
<html>
<head>
<title>this 关键字的使用</title>
</head>
<body>
<script language="JavaScript" type="Text/JavaScript">
//Class of ActNode
function ActNode()
{
    this.id ;
    this.setValue = function(_id) //必须写成这个样子，不能写成 function setValue
    ()
    {
    this.id = _id;
    }
}
var obj = new ActNode();
obj.setValue("关键字 this 的使用");
alert(obj.id);
</script>
</body>
</html>
```

在上述代码中，首先创建 ActNode()函数，在该函数中创建一个 setValue 方法。然后使用 this 关键字引用 setValue 方法，在该方法中传递一个形式参数，并将传递的参数值赋给 id。接着实例化一个对象 obj，使用 obj 调用 setValue 方法为其赋值。最后使用一个对话框的形式将 id 的值输出。

11.6　对象

JavaScript 语言是基于对象，而不是面向对象。之所以是一门基于对象的语言，主要是因

为其没有提供抽象、继承和重载等有关面向对象的语句功能，而是将其他语言所创建的复杂对象统一起来，从而形成一个非常强大的对象系统。

11.6.1　对象概述

JavaScript 中的对象是由属性（properties）和方法（methods）两个基本元素构成的。属性在对象实施其所需要的行为过程中，实现信息的存储，通常与变量相关联。方法是指对象能够按照开发者的意图而被执行，通常与特定的函数相关联。

使用运算符 new 创建对象，在这个运算符之后必须有初始化对象的构造函数名。例如使用如下方式可以创建一个空对象：

```
var obj = new Object();
```

对象创建好以后，就可以引用对象。对象被引用之前，这个对象必须存在，否则应用将毫无意义，而且会出现错误信息。一个对象要真正地被使用，可以采用以下几种引用对象的方式。

- 引用 JavaScript 内置对象;
- 由浏览器环境提供;
- 创建新对象。

1. 有关对象操作的语句

JavaScript 不是面向对象的语言，它没有提供面向对象语言的许多功能，因此，JavaScript 设计者将其称为基于对象而不是面向对象的语言。在 JavaScript 中提供了几个用于操作对象的语句和关键字及运算符。

- **for in 语句**

使用 for in 语句操作对象的语法如下所示：

```
for （变量 in 对象）
{
   语句块;
}
```

该语句的功能是对已知对象的所有属性进行循环操作。是将一个已知对象的所有属性反复地赋给一个变量，而不是使用计数器实现。使用该语句的优点就是无需知道对象中属性的个数即可操作。

- **with 语句**

在 with 语句块内，任何对变量的引用都会被认为是这个对象的属性，使用 with 语句操作对象的语法如下所示：

```
with(对象)
{
   语句块;
}
```

□ **this 关键字**

this 是对当前的引用，在 JavaScript 中由于对象的引用是多层次、多方位的，往往对一个对象引用时又需要引用另一个对象，而另一个对象有可能又要引用其他对象，这样有可能造成混乱，最后也不知道现在引用的是哪一个对象。为此 JavaScript 提供 this 关键字用于指定当前对象的语句。

□ **new 运算符**

虽然在 JavaScript 中对象的功能已经非常强大。但更强大的是设计人员可以按照需求自定义对象，以满足某一特定的要求。使用 new 运算符可以创建一个新的对象。

2．对象属性的引用

对象属性的引用可以通过 3 种方法实现，分别是使用点（.）运算符、通过对象的下标和字符串的形式。

□ **使用点（.）运算符**

```
myPeople.Name="Lee Anne";
myPeople.Age="21";
myPeople.Sex="女";
```

在上述代码中 myPeople 是一个已经存在的对象，Name、Age 和 Sex 分别是 myPeople 的属性，并通过操作对其赋值。

□ **通过对象的下标**

```
myPeople[0]="Lee Anne";
myPeople[1]="21";
myPeople[2]="女";
```

对象的属性定义好以后，可以通过数组的形式访问属性。访问操作如下所示：

```
function people(object)
{
  for(var i=0;i<3;i++)
  document.write(object[i]);
}
```

□ **通过字符串的形式**

```
myPeople["Name"]="Lee Anne";
myPeople["Age"]="21";
myPeople["Sex"]="女";
```

在数组的下标中使用字符串进行标识，使用时也可以通过字符串来引用。

3．对象方法的引用

JavaScript 中对象方法的引用非常简单，其语法如下所示：

```
objectName.methods();
```

methods()方法实际上就是一个函数。例如引用 myPeople 对象中的 people()方法，则可以使用如下语句：

```
document.write(myPeople.people());
```

11.6.2 内置对象

在 JavaScript 中对于对象属性和方法的引用有两种情况，一种是静态对象，即在引用该对象的属性或方法时不需要创建实例；另一种在引用时必须为其创建一个实例，即动态对象。

对于 JavaScript 内置对象的引用，也是紧紧围绕着属性与方法进行的。因而明确对象的静态和动态对于掌握和理解 JavaScript 内置对象十分有帮助。

1. String 对象

在 JavaScript 中使用 String 对象处理字符串。String 是 JavaScript 最重要的核心对象之一，也是在编程过程中使用最多的对象。

当声明一个变量并赋予字符串类型的值时，说明已经创建一个 String 对象。例如：

```
var myname="Lee Anne";
```

在该语句中，首先创建一个 myname 变量，然后将一个字符串对象分配给这个变量，这与给变量赋值相同。下面，使用另一种创建 String 对象的方法，该方法是将一个字符串作为参数调用。例如：

```
var myname= new String("Lee Anne");
```

任何一个字符常量都是一个 String 对象，可以将其直接作为对象使用。但是这两种创建 String 对象方法的主要区别就是返回类型不同。使用 new String()创建返回的是一个 Object，而直接赋值返回的则是一个 String。

表 11-8 概述 String 对象具有的一些属性和方法，并进行简要说明。

表 11-8 String 对象的属性和方法

属性和方法	说明
length 属性	属性值是一个只读整数，用于获取 String 对象中字符的数量
prototype 属性	运行开发者为 JavaScript 内置对象添加的方法和属性
charAt(index)方法	返回字符串中 index 处的字符
substring(indexA,indexB)方法	获取自 indexA 到 indexB 的子字符串
lastIndexOf(searchValue[,fromIndex])方法	从字符串的尾部向前搜索 searchValue，并报告找到的第一个实例
indexOf(searchValue[,fromIndex])方法	在字符串中查找第一次出现的 searchValue。如果给定 fromIndex，则从字符串内该位置开始查找，当找到 searchValue 后，返回该字符串的第一个字符的位置
toLowerCase()方法	将字符串中所有字符转换为小写
toUpperCase()方法	将字符串中所有字符转换为大写

在了解 String 对象的方法和属性之后，下面通过示例说明 String 对象的方法和属性的使用，具体如代码 11.20 所示。

代码 11.20　String 对象

```
<html>
<head>
<title>String 对象</title>
</head>
<body>
<h3>使用 indexOf()方法</h3>
<script>
var userEmail = prompt("Please enter you email address","");
document.write(userEmail.indexOf("@"));
document.write("<br/>");
//使用 length 属性
document.write("你输入的邮箱地址的长度是:"+userEmail.length);
</script>
<h3>使用 substring()方法</h3>
<script>
var oldstr="My name is Anne";
var newstr=oldstr.substring(1,6);
document.write(newstr);
</script>
</body>
</html>
```

上述代码中，在第一个<script>标签中，首先使用 prompt()方法获取输入的 E-mail 地址，然后使用 indexOf()方法，检查输入的 E-mail 中是否包含有@符号，如果没有发现@，在页面中则会输出一个–1。只要这个字符出现在输入字符串中的任何位置，就会返回索引中它的位置，也就是说返回大于–1 的数。接着再使用 length 属性获取输入的 E-mail 地址的长度。在第二个<script>标签中，使用 substring()方法设置字符串的起始和结束位置的索引作为参数，从字符串中截取一个字符串。

2. Array 对象

数组提供一种理想的方法存储、操作、排序和获取数据集，被定义为一组有某种共同特性的元素，例如相似性和类型的集合。集合中的所有元素之间由一个特殊的标识符来区分，该标识符通常被称为键（Key）或下标。在 JavaScript 中，使用 Array 对象对数组进行操作。创建 Array 对象的语法如下：

```
var 数组名 = new Array();
```

定义好数组以后，就可以向数组中添加元素，例如：

```
var myPeople = new Array(); //定义一个名称为 myPeople 的数组
myPeople[0] ==" Anne";      //添加元素
myPeople[1] = "Lee";
```

当向数组中添加元素时，方括号（[]）不可以省略，因为数组的下标表示方法是用方括号括起来的。如果在定义数组时直接初始化，可以使用如下方式：

```
var 数组名 = new Array(元素 1,元素 2…);
```

Array 对象也提供了一些方法和属性，首先介绍 Array 对象的一些属性，如表 11-9 所示。

表 11-9　Array 对象的属性

属性	说明
index	字符在字符串中的匹配位置，如果找不到则返回–1
input	指定匹配正则表达式的原始字符串
length	返回数组的长度，等于数组中的最后一个元素的下标加一。其数值会随着数组元素的增减而自动改变
Prototype	该属性是所有 JavaScript 对象所共有的属性，其作用是将新定义的属性和方法添加到 Array 对象中

length 表示的是数组所占内存空间的数目，而不仅仅是数组中元素的个数。下面通过示例理解 length 属性的使用，如代码 11.21 所示。

代码 11.21　length 属性的使用

```
<html>
<body>
<script>
var aa = new Array("xiaoming","xiaohuang","xiaohua");
delete aa[1]; //删除一个元素
document.write(aa.length);
</script>
<script>
var aa = new Array("xiaoming","xiaohuang","xiaohua");
aa[aa.length]=aa.length+1; //增加一个元素
aa.length=1;
document.write(aa.length);
</script>
</body>
</html>
```

保存文件，在浏览器中查看运行结果。可以看到删除一个元素，在页面中显示的结果还是 3，这说明即使删除元素也无法改变数组的长度。当数组的长度等于 1 时，即使在这个数组中包含有很多元素，但在页面中显示的结果是 1，说明在该数组中只剩下一个元素。

对 Array 对象的属性了解之后，接下来介绍 Array 对象的方法，如表 11-10 所示。

表 11-10　Array 对象的方法

方法名称	说明
concat()	用于对象连接引用、字符或字符串，复制其值
join()	由指定的分隔符分割所有元素
pop()	从数组中移除最后一个元素并将该元素返回，如果该数组为空，则返回 undefined
push()	将新元素按出现的顺序追加，如果参数是数组，则该数组将作为单个元素添加到数组中
shift()	将数组中的一个元素移除并返回

续表

方法名称	说明
unshift()	将元素插入到一个数组的开头，以便按其在参数表中的次序排列
reverse()	将数组元素按照与原来相反的方向重排
sort()	返回排序后的 Array 对象，默认元素将按 ASCII 字符顺序的升序进行排序
splice()	通过移除从 start 位置开始的指定个数的元素并插入新元素修改数组
TolocalString()	以字符串的形式返回一个值，该值适合于当前环境的区域设置
ToString()	返回表示对象的字符串
valueOf()	返回指定对象的原始值

Array 对象方法的调用与其他对象一样，在实际应用时，数组多数用于排序，例如从小到大排序或者从大到小排序。下面通过示例理解 sort()方法的使用，如代码 11.22 所示。

代码 11.22　sort()方法的使用

```
<html>
<body>
<script type="text/javascript">
function AscSort(x, y) {
   return x == y ? 0 : (x > y ? 1 : -1);
}
function DescSort(x, y) {
   return x == y ? 0 : (x > y ? -1 : 1);
}
function RandomSort(x, y) {
   return Math.floor(Math.random() * 2 - 1 );
}
var array = [2,4,3,5,1,6,9,0,8];
document.write("<p>正序: " + array.sort(AscSort) + "</p>");
document.write("<p>倒序: " + array.sort(DescSort) + "</p>");
document.write("<p>随机排序: " + array.sort(RandomSort) + "</p>");
document.write("<p>随机排序: " + array.sort(RandomSort) + "</p>");
document.write("<p>随机排序: " + array.sort(RandomSort) + "</p>");
</script>
</body>
</html>
```

上述代码中，定义了 3 个函数，然后分别调用这 3 个函数，对数组进行排序。保存文件，在浏览器中查看运行效果，如图 11-13 所示。

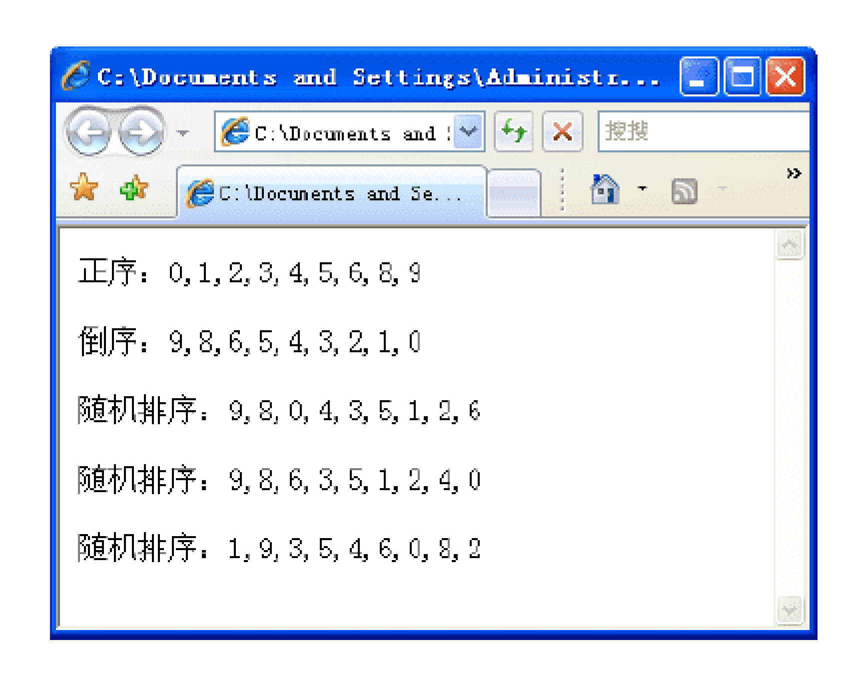


图 11-13　sort 方法的使用

3. Math 对象

Math 对象的作用是执行数学计算，提供标准的数学常量和函数库，将常量定义为 Math 的属性，函数定义为 Math 的方法，而且在使用这个对象之前不需要创建实例，因为 Math 是内置对象而不是对象类型。

表 11-11 列出了 JavaScript 提供的 8 种 Math 对象定义的属性。

表 11-11　Math 属性

名称	Math 方法	名称	Math 方法
常数	Math.E	2 的自然对数	Math.LN2
圆周率	Math.PI	10 的自然对数	Math.LN10
2 的平方根	Math.SQRT2	以 2 为底的 e 的对数	Math.LOG2E
1/2 的平方根	Math.SQRT1_2	以 10 为底的 e 的对数	Math.LOG10E

表 11-12 列出了 Math 对象定义的方法。

表 11-12　Math 方法

方法名称	说明
abs(x)	返回 x 的绝对值
acos(x)	返回 x 的反余弦值（余弦值等于 x 的角度），用弧度表示
asin(x)	返回 x 的反正弦值
atan(x)	返回 x 的反正切值
atan2(x,y)	返回复平面内点（x,y）对应的复数的辐角，用弧度表示，其值在–π 到 π 之间
ceil(x)	返回大于等于 x 的最小整数
cos(x)	返回 x 的余弦
exp(x)	返回 e 的 x 次幂（e^x）
floor(x)	返回等于小于 x 的最大整数
log(x)	返回 x 的自然对数（ln x）
max(a,b)	返回 a、b 中较大的数
min(a,b)	返回 a、b 中较小的数
pow(m,n)	返回 n 的 m 次幂（n^m）
round(x)	返回 x 四舍五入的值
random()	返回大于 0 小于 1 的一个随机数
sin(x)	返回 x 的正弦
sqrt(x)	返回 x 的平方根
tan(x)	返回 x 的正切

在对 Math 对象的方法做一些简单的了解之后，接下来通过示例说明如何使用该对象的方法，具体如代码 11.23 所示。

代码 11.23　round、floor 和 ceil 方法的使用

```
<html>
<body>
<script>
var numberToRound = prompt( "Please enter a number", "" )
document.write( "round( ) = " + Math.round( numberToRound ) );
document.write( "<br/>" );
document.write( "floor( ) = " + Math.floor( numberToRound ) );
document.write( "<br/>" );
document.write( "ceil( ) = " + Math.ceil( numberToRound ) );
</script>
```

```
</body>
</html>
```

上述代码中，使用 prompt()方法获取用户输入的一个值。运行程序，例如输入 22.65，在页面中输出的结果如下所示：


```
round( ) = 23
floor( ) = 22
ceil( ) = 23
```

4．Date 对象

Date 对象以毫秒为单位表示特定的时间段。如果某个参数的值大于其范围或为负数，存储的其他值将做相应的调整。

在 JavaScript 中为 Date 对象提供了一系列的方法，如表 11-13 所示。

表 11-13　Date 对象方法

方法名称	说明
getDate()	返回 Date 对象是当月的第几天
getDay()	返回 Date 对象对应本周的星期几
getHours()	返回 Date 对象的小时数
getMinutes()	返回 Date 对象的分钟数
getMonth()	返回 Date 对象的月份
getSeconds()	返回 Date 对象的秒
getTime()	返回 Date 对象的时间
getTimeZoneOffset()	返回 Date 对象的时区偏差（以分钟为单位）
getYear()	返回 Date 对象的年份
parse()	返回自本地时间 1970 年 1 月 1 日以来的毫秒数
setDate(integer)	设置 Date 对象为当月的第几天
setHour(integer)	设置 Date 对象的小时数
setMinutes(integer)	设置 Date 对象的分钟数
setMonth(integer)	设置 Date 对象的月份
setSeconds(integer)	设置 Date 对象的秒
setTime(integer)	设置 Date 对象的时间
setYear(integer)	设置 Date 对象的年份
toGMTString()	将日期转换为 GMT 格式字符串
toLocalString()	将日期转换为本地格式，即用户所在地区常用的格式
UTC()	返回自 1970 年 1 月 1 日午夜（GMT）起的毫秒数

此外，Date 对象还对用户提供多种类型的构造函数，使其在实例化时就可以创建对象，表 11-14 中列出这些构造函数的形式及说明。

表 11-14　Date 构造函数

构造函数形式	说明
Date()	用当前时间和日期创建 Date 实例
Date(dateString)	用 dateString 参数指定日期创建 Date 实例。dateString 的格式为"month day,year hours:minutes:seconds"

续表

构造函数形式	说明
Date(year,month,day)	用整数的年、月、日值指定日期来创建 Date 实例，年参数为 0 到 99
Date(year,month,day,hours,minutes,seconds)	用整数的年、月、日、时、分、秒值指定日期来创建 Date 实例，年参数为 0 到 99

有了 Date 对象可以很容易地对日期和时间进行获取、运算和判断等。下面通过示例讲解 Date 对象的使用方法，具体如代码 11.24 所示。

代码 11.24　Date 对象的使用

```
<html>
<body>
<script type="text/javascript">
function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
//当数字小于 10 的时候在前面加一个 0
m=checkTime(m);
s=checkTime(s);
document.getElementById('clock').innerHTML=h+":"+m+":"+s;
//每隔 500 毫秒重新执行一次
startTimet=setTimeout('startTime()',500);
}
function checkTime(i)
{
if (i<10)
 {
  i="0" + i;
 }
  return i;
}
</script>
  <div id="clock"></div>
  <input type="button" name="button" onclick="startTime(); return false" value
  ="启动时钟">
 </body>
</html>
```

保存文件之后，在浏览器中查看运行效果。当用户单击【启动时钟】按钮时，将获取系统的时间显示在 div 标签中，并且时间不断更新。

11.6.3　浏览器对象

在 HTML 中，使用 JavaScript 的一个好处就是可以控制 Web 文档及其内容。JavaScript 脚本将一个新页面载入浏览器，操作浏览器的窗口和文档、打开新窗口以及动态修改页面的内容。为了操作浏览器和文档，JavaScript 使用分层的父对象和子对象，这称为 DOM（Document Object Model，文档对象模型）。这些对象的组织类似于一个树形结构，并表示一个 Web 文档的所有内容和组件，如图 11-14 所示。

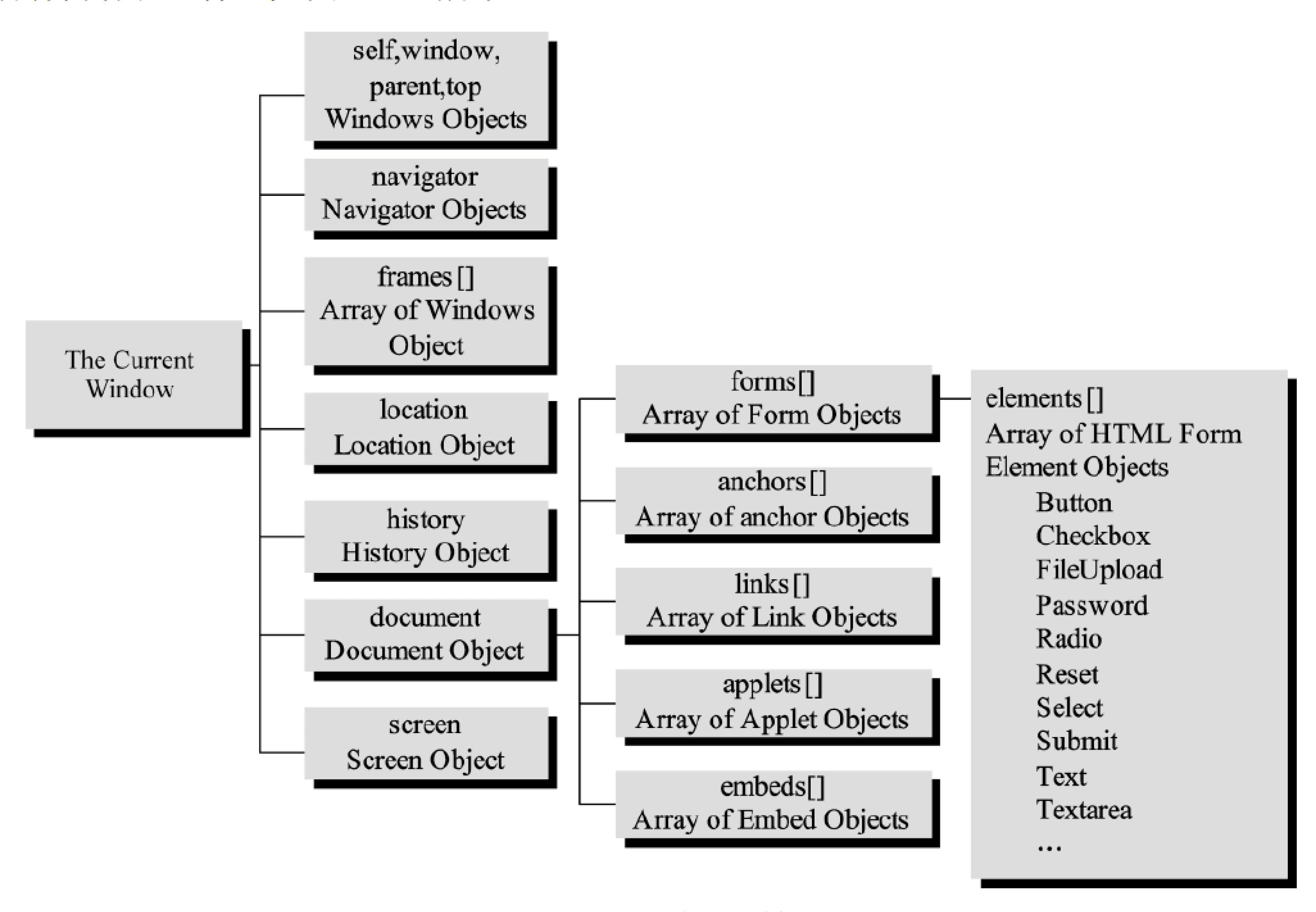


图 11-14　图像层次模型

Window 对象是客户端程序的全局对象，也是客户端层次的根。每个浏览器窗口以及窗口中的框架都由 Window 对象表示。Window 对象定义许多属性和方法。在表 11-15 中列出 Window 对象常用的方法。

表 11-15　Window 对象的方法

方法名称	说明
alert()	弹出一个只包含【确定】按钮的对话框
confirm()	弹出一个包含【确定】和【取消】按钮的对话框，要求用户做出选择。如果用户单击【确定】按钮，则返回 true 值，如果单击【取消】按钮，则返回 false 值
prompt()	弹出一个包含【确定】和【取消】按钮，以及一个文本框的对话框，要求用户在文本框中输入一些数据
close()	关闭窗口
open()	打开新的窗口，可以指定新窗口的各种属性，如 URL 等
focus()、blur()	请求或放弃窗口的键盘焦点。focus()方法把窗口置于最上层，使窗口可见

11.6.4 自定义对象

虽然 JavaScript 语句是基于对象的，但还是具有一些面向对象的基本特征。它可以根据自身的需要创建自定义对象，从而进一步扩大 JavaScript 的应用范围，增强编写功能强大的 Web 文档的能力。

如果在 JavaScript 中使用一个新的对象，必须先定义一个对象，然后再为该对象创建一个实例。这个实例就是一个新的对象，具有对象定义的基本特征。自定义对象的基本格式如下所示：

```
function Object(属性表)
{
  this.prop1 = prop1;
  ……
  this.meth = functionName1;
  ……
}
```

下面通过示例说明如何定义一个自定义对象，可以为该对象指明其属性和方法，通过属性和方法构成一个对象的实例，如代码 11.25 所示。

代码 11.25　自定义对象

```
<html>
<head>
</head>
<script language="javascript">
function printcolor()
{
 document.write("this apple's color is "+this.color+"<br/>");
}
function printsize()
{
 document.write("this apple's size is "+this.size+"<br/>");
}
function apple(tcolor,tsize)        //自定义对象 apple 的构造函数
{
 this.color=tcolor;                 //color 属性
 this.size=tsize;                   //size 属性
 this.pcolor=printcolor;            //pcolor 方法
 this.psize=printsize;              //psize 方法
}
var apple1=new apple("red","big");  //定义一个 apple 对象的实例，并进行了初始化
var apple2=new apple("green","small");
apple1.pcolor();                    //调用对象方法
```

```
apple1.psize();
apple2.pcolor();
apple2.psize();
</script>
</html>
```


保存程序，在浏览器中查看运行效果，如下所示：

```
this apple's color is red
this apple's size is big
this apple's color is green
this apple's size is small
```

第12章 XML 编程基础



内容摘要 | Abstract

XML 是一种描述性语言，用于对网络数据的转换和描述。允许用户使用自定义标记来描述 XML 文档所要表达的内容。虽然 XML 允许用户自定义标记，但是必须遵循 XML 规范及文档结构。随着 XML 的发展，出现了许多与之相关的技术，如 DTD、XSLT 和 DOM 等。使用 XML 文档需要熟悉其文档结构、创建语法、命名空间等。在文档中使用 DTD 或者 XML Schema 架构进行元素和属性的声明，可以提高文档的规范性和有效性。

学习目标 | Objective

- 理解并掌握 XML 的基本语法
- 掌握 DTD 在 XML 文档中定义的方式
- 掌握 DTD 对元素和属性的声明
- 理解 DTD 实体的定义
- 掌握 XML Schema 的数据类型
- 了解 XSLT 的工作原理
- 掌握 XSLT 元素的用法

12.1 XML 基本概念

XML 与 HTML 一样，都是属于 SGML（标准通用语言）。XML 是 Internet 环境中跨平台的、依赖于内容的技术，是当前处理结构化文档信息的有力工具。XML 是一种简单的数据存储语言，使用一系列简单的标记描述数据，而这些标记可以很方便地建立。虽然 XML 比二进制数据占用更多的空间，但 XML 更易于掌握和使用。

12.1.1 XML 简介

XML 是可扩展标记语言（eXtensible Markup Language），即是可扩充的。这意味着可以是自定义的标记集，是一种元语言，可以用于描述其他语言。

XML 可以深入到分布式应用、数据库应用领域。通过 XML 可以将业务逻辑和显示分开。通过 XML 的标记来表示数据的逻辑结构，这样就保持了数据库的一些相关特性，例如数据查

询等，而且很容易进行相应的程序开发。然后再通过程序将处理后的 XML 转化为响应显示风格的 HTML，其中最大的优势就是中间代理层可以集合不同的数据库和现有的文件数据。

XML 可以用于分离用户查看数据的界面，使用简单灵活开放的格式可以给 Web 创建功能强大的应用软件。XML 文档的结构可以分为 3 个部分：序言、主体和尾声。其中，每个 XML 文档必须保留序言和主体，而尾声可以根据用户的需要来使用。

1. 序言

XML 文档的序言包括 XML 的声明和注释两个部分。声明一个 XML 文档需要指明所有的 XML 版本和文档编码等信息。注释用于描述 XML 文档的特征及内容，方便用户在使用中对文档的理解和阅读。一个规范的 XML 文档以 XML 声明作为文档的开始。

下面是一个完整的 XML 声明：

```
<?xml version = "1.0" encoding = "Gb2312" standalone = "yes/no"?>
<!--注释-->
```

在 XML 声明中各参数的含义如下所示。

- **version**　指定采用的 XML 版本号，而且必须在属性列表中第一个出现。
- **encoding**　指定文档的编码格式。常用的编码格式有 UTF-8 和 GB2312 两种。
- **standalone**　表示使用 DTD 的形式。值只能是 yes 或 no，默认值为 no。如果为 yes，说明所有实体声明都包含在文档中；如果是 no，说明需要引用外部实体。

2. 主体和尾声

主体就是 XML 文档描述数据的地方，通常由标记、元素和属性等组成。XML 文档的尾声包括注释、处理指令或者紧跟元素树后的空白。由于大多数应用程序在文档根元素的结束标记处就结束，而不再对尾声进行任何处理，所以尾声对于 XML 文档不起任何作用。用户可以根据需要选择是否添加尾声。

12.1.2　XML 标记、元素和属性

XML 是一种基于文本的标记语言。有 3 个通用术语用于描述 XML 文档的组成部分：标记、元素和属性，它们是构成 XML 文档的重要组成部分。如代码 12.1 所示，说明这些术语及其使用。

代码 12.1　XML 示例

```
<?xml version="1.0" encoding="GB2312"?>
<address>
  <name>
    <title>Hello XML</title>
    <first-name>
     Anne
    </first-name>
```

```
    <last-name>
     Lee
    </last-name>
  </name>
  <street>
    207 Main Street
  </street>
  <city state="zz">zhengzhou</city>
  <postal-code>
    454000
  </postal-code>
</address>
```

- 标记是左尖括号“<”和右尖括号“>”之间的文本。有开始标记（例如<name>）和结束标记（例如</name>）。
- 元素是开始标记、结束标记以及位于两者之间的所有内容。在上述代码中，<name>元素包含 3 个子元素：<title>、<first-name>和<last-name>。
- 属性是一个元素的开始标记中的名称-值对。在该示例中，state 是<city>元素的属性。



在<>内部和标记间都支持中文，但是建议不要使用中文做标记，这样容易导致许多错误和不便。

1. 标记

在 XML 中，标记可以分为非空标记和空标记两种。非空标记用尖括号“<>”表示，以左尖括号“<”开始，右尖括号“>”结束。空标记以左尖括号“<”开始，以“/>”结束。空标记不包含任何内容，所以空标记没有开始和结束的标记，但是空标记中可以包含属性，如下代码所示：

```
<name/><!--空标记-->
<name first-name = "Anne" last-name = "Lee" /><!--空标记-->
<!--非空标记-->
<name>
<first-name>Anne</first-name>
<last-name>Lee</last-name>
</name>
```



无论是非空标记还是空标记，在左尖括号“<”前不能有空格。

2. 元素

元素是 XML 文档中最重要的部分。元素用于描述其所包含的数据，还可以包含属性名称

和值用于提供有关内容的其他信息，并指出这些信息的逻辑结构。在 XML 文档中只有一个根元素，其他元素在根元素内以树形分层结构排列，而且元素可以嵌套使用。

在 XML 中，元素也分为非空元素和空元素两种。非空元素由开始标记、结束标记和两标记之间的数据构成。标记之间的数据被认为是元素的值。非空元素的语法结构是：

```
<name>Anne Lee</name>
```

而空元素就是不包含任何内容的元素，其语法结构是：

```
<name></name>
```

在 XML 解析器中对空元素和空标记的处理相同，因为空元素与空标记的格式一样。在 XML 文档中，规范的 XML 标记与元素更有助于正确描述数据，所以必须遵循如下的命名规则。

- 元素名中可以包含字母、数字和其他字符，如<address>、<学号>等。元素名中虽然可以包含中文，但是有一些软件不能很好地支持这种命名，所以尽量使用英文字母命名。也不要使用“:”字符，因为 XML 命名空间需要用到这个十分特殊的字符。
- 标记或元素名必须规范，名称必须以字母、下划线“_”或中文开头，而不能用数字或标点符号开头，并且中间不允许有空格。
- 文档中的标记必须对应，每一个 XML 文档都必须有开始标记和结束标记。
- 标记大小写敏感，例如<street>和<Street>是两个不同的标记。
- 元素标记必须合理的包含，在 XML 文档中不允许不合理的嵌套包含，即开始标记和结束标记要紧紧呼应。
- 元素中的开始标记和结束标记要保证成对出现，并且大小写一致。

3. 属性

XML 属性可以将一些额外的信息附加到元素上，从而使文档对元素数据特性的描述更加具体。如果不希望通过子元素描述元素的一些特性，则可以使用属性。属性一般在元素的开始标记中声明，由属性名和值构成。在非空元素和空元素中声明属性的语法格式如下：

```
非空元素 <标记名 属性列表>数据内容</标记名>
空元素 <标记名 属性列表></标记名>
      或<标记名 属性列表/>
```

下面通过示例讲解属性在 XML 中的用法。例如描述一个学生的详细信息，如代码 12.2 所示。

代码 12.2　XML 的属性

```
<?xml version="1.0" encoding="GB2312"?>
<Students>
<student name="bob" id="00102" class="一年级一班">学生信息</student>
<student name="jojo" id="00103" class="一年级一班"></student>
</Students>
```

在 XML 文档中，使用子元素和属性都可以实现对数据描述信息的存储。例如下面的 XML 文档使用子元素描述，如代码 12.3 所示。

代码 12.3　使用子元素描述信息

```
<?xml version="1.0" encoding="GB2312"?>
<Students>
<student>
    <name>bob</name>
    <id>00102</id>
    <class>一年级一班</class>
</student>
<student>
    <name>jojo</name>
    <id>00103</id>
    <class>一年级一班</class>
</student>
</Students>
```

在 XML 文档中，对于何时使用属性、何时使用子元素并没有特定的规定，只能根据执行环境和需要来决定使用哪个实现方式。但是在文档中使用属性会引发下列问题。

- 属性不能包含多个重数值（子元素可以）。
- 属性不容易扩展。
- 属性不能够描述文档结构（子元素可以）。
- 属性很难被程序代码处理。
- 属性值很难通过 DTD 进行测试。

所以建议尽量使用子元素而少用属性。由于 XML 的高扩展性，所以数据在 XML 中需要经常增加或修改。如果将增加或修改的数据放入到属性中，那么将非常不利于对数据的维护和更新。

12.1.3　XML 命名空间

XML 命名空间提供一种避免元素命名冲突的方法，因为 XML 文档中使用的元素不固定，那么两个不同的 XML 文档使用同一个名字描述不同类型元素的情况就可能发生。XML 命名空间由前缀和本地部分组成，中间用冒号“：”隔开。命名空间的属性一般放置在元素的开始标记处，其声明语法如下所示：

```
<xmlns:namespace-prefix="namespace">
```

在 XML 命名空间声明语法中，xmlns 是必须属性；namespace-prefix 表示命名空间的别名，其值不能为 XML；namespace 表示标识抽象或物理资源的紧凑字符串。

下面通过示例说明如何声明和使用命名空间，具体实现如代码 12.4 所示。

代码 12.4　命名空间

```
<?xml version="1.0" encoding="GB2312"?>
<Students xmlns:addr="http://www.itzce,com/address"
        xmlns:books="http://www.xxx.com/books">
<addr.student>
   <name>bob</name>
   <id>00102</id>
   <class>一年级一班</class>
</addr.student>
<books.student>
    <name>jojo</name>
   <id>00103</id>
   <class>一年级一班</class>
</books.student>
</Students>
```

在该示例中声明两个命名空间前缀分别为 addr 和 books。在这里为特定元素定义命名空间意味着该元素的所有子元素都属于同一个名称空间。第一个<name>元素属于 addr 命名空间，因为其父元素<addr:student>属于该命名空间。

12.1.4　XML 实体引用及 CDATA 段

如果在 XML 文档中放置一个类似“<”字符，那么这个文档会产生一个错误，这是因为解析器会将其解释为新元素的开始。但是在 XML 文档中难免会用到一些特殊字符，例如左尖括号“<”、右尖括号“>”、连接符号“&”、单引号“'”或者双引号“"”，这些字符属于 XML 文档的内置字符，如果需要将这些字符在 XML 文档中显示出来，就要使用实体引用。在 XML 中有 5 个预定义的实体引用，如表 12-1 所示。

表 12-1　XML 实体引用

实体引用	特殊符号	含义	实体引用	特殊符号	含义
<	<	小于号	'	'	单引号
>	>	大于号	"	"	双引号
&	&	和或连接符			

一般情况下 XML 解析器会对 XML 文档中出现的任何文本内容进行处理。但是如果有些文本内容在 CDATA 段内，那么就会被 XML 解析器忽略。CDATA 段是用于包含文本的方法，它通常用于建立代码的脚本，例如 JavaScript。

CDATA 段以“<![CDATA[”开始，以“]]>”结束。开始区段与结束区段之间为所要包含文本的内容。CDATA 段的基本语法格式如下所示：

```
<![CDATA[…]]>
```

下面通过示例说明如何在 XML 文档中引用实体和使用 CDATA 段，具体实现如代码 12.5

所示。

代码 12.5　引用实体和使用 CDATA 段

```
<?xml version="1.0" encoding="GB2312"?>
<project>
<message>if salary &lt; 1000 then</message>
<script>
<![CDATA[
function matchwo(a,b)
{
if (a < b && a < 0)
   {
   return 1
   }
else
   {
   return 0
   }
}
]]>
</script>
</project>
```

将上述代码保存为“引用实体.xml”文件，运行结果如图 12-1 所示。可以看到引用“<”实体在页面中输出的是一个“<”；在 CDATA 段中的内容不进行解析，而是原样输出。

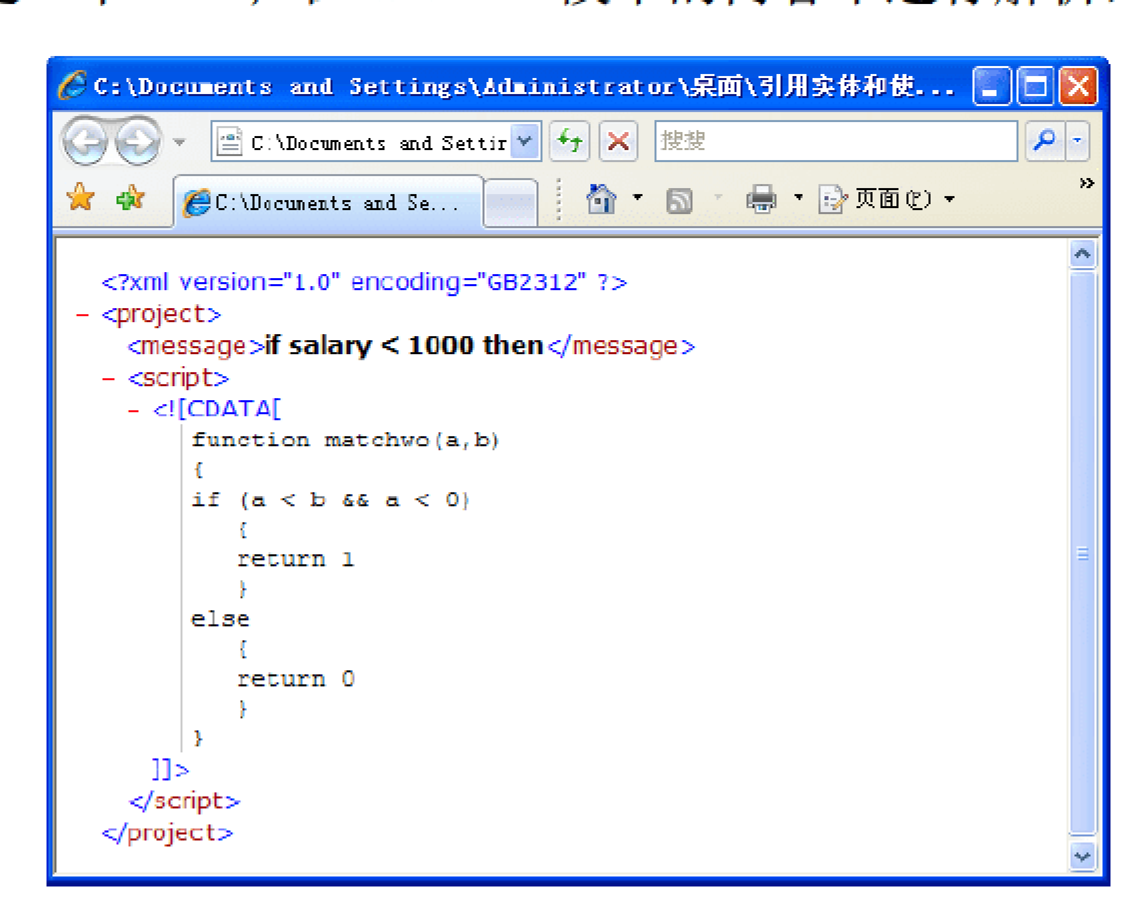


图 12-1　引用实体和使用 CDATA 段

12.2　文档类型定义 DTD

在 XML 文档中使用 DTD 可以保证该文档中的元素或属性声明的有效性，从而保证 XML

文档的有效性。本节首先介绍 DTD 的基本概念及结构，接着介绍通过 DTD 进行元素和属性的声明，最后介绍 DTD 实体的声明和引用。

12.2.1　DTD 简介

DTD（Document Type Definition，文档类型定义）可定义合法的 XML 文档构建模块，使用一系列合法的元素定义文档的结构。一个 DTD 文档包含元素的定义规则、元素间关系的定义规则、元素可使用的属性、可使用的实体或符号规则。

DTD 文件提供一组元素和属性供工作组成员使用，还定义有关元素在结构层次中的显示位置的规则。例如 DTD 文件可能要求 Title 元素作为 Story 元素的子元素，因为标题应该显示在文章内部；如果为标题添加标签，但是却没有为显示标题的文章添加标签，那么 DTD 文件将把 Title 元素标记为无效元素。使用 DTD 文件可以在 InDesign 文件中搜索并标出无效结构错误，这个过程称为验证。

DTD 根据其出现的位置可分为内部 DTD 和外部 DTD 两种。内部 DTD 是指 DTD 和相应的 XML 文档都在同一个文档中。如果 DTD 需要在不同的 XML 文档共享，或者为了将来更好、更方便地更新文档数据，推荐使用外部 DTD。外部 DTD 是在 XML 文档之外创建扩展名为.dtd 的文件，这样 DTD 就可以实现多个 XML 文档共享。

12.2.2　内部 DTD 和外部 DTD

内部 DTD 是在 XML 文档中直接设定 DTD。外部 DTD 是已经编辑好的，并且可以被不同的 XML 文档共享和调用的 DTD。在 XML 中 DTD 由文档类型声明引入，文档类型声明以“<!DOCTYPE”开始，以“]>”结束。通常将开始和结束放在不同的行上，并允许中间有空格或断行。

如果 XML 文档中的 standalone 属性值为 yes，说明没有相应的外部 DTD 文件存在；如果值为 no，则可能存在这样一个文件。即使"standalone=no"也可能没有外部 DTD 存在。

1. 内部 DTD

内部 DTD 又称为 DTD 的内部子集，可以在 XML 文档中直接定义。内部 DTD 的声明如下所示：

```
<!DOCTYPE 根元素名 [
<!ELEMENT 根元素名 (子元素列表)>
<!ELEMENT 子元素名 EMPTY / ANY / #PCDATA /(子元素内容)/(混合内容)>
<!ATTLIST 元素名 属性名 属性类型 属性默认值>
]>
```

下面通过示例说明如何在 XML 文档中定义内部 DTD，具体实现及使用如代码 12.6 所示。

代码 12.6　内部 DTD

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
```

```
<!DOCTYPE student [
  <!ELEMENT student (name,id,class,body)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT id (#PCDATA)>
  <!ELEMENT class (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
<student>
    <name>bob</name>
    <id>00102</id>
    <class>一年级一班</class>
    <body>Don't forget the meeting this weekend!</body>
</student>
```

在上述代码中，DTD 的解释如下所示。

- !DOCTYPE student（第二行）定义此文档是 student 类型的文档。
- !ELEMENT student（第 3 行）定义 student 元素有 4 个子元素：name、id、class 和 body。
- !ELEMENT name（第 4 行）定义 name 元素为#PCDATA 类型。

将上述代码保存为“内部 DTD.xml”，然后在浏览器中的运行结果如图 12-2 所示。

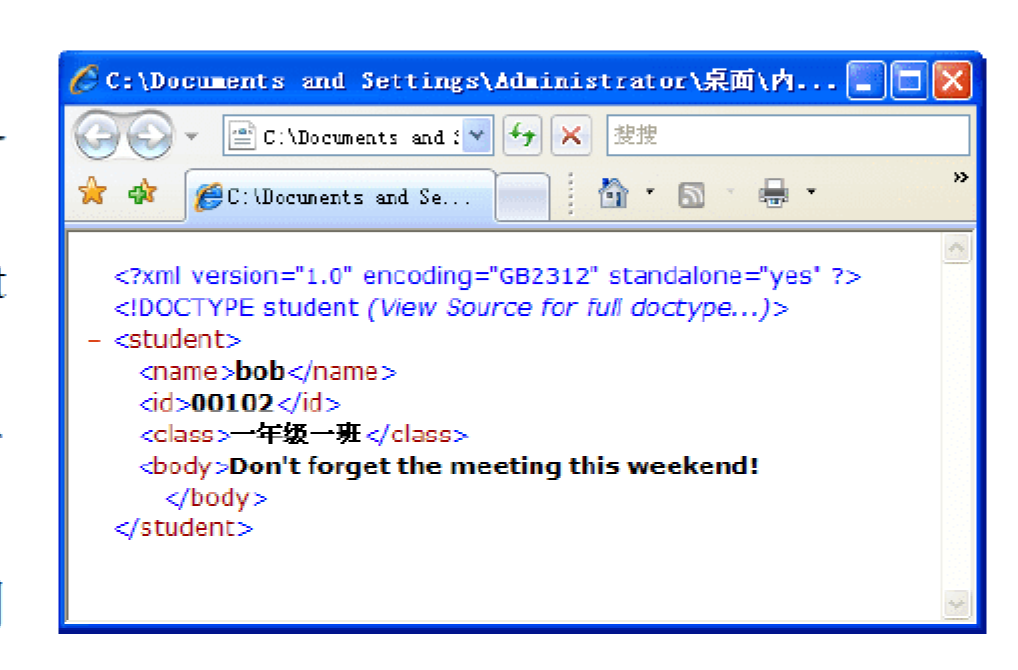


图 12-2　定义内部 DTD

从运行结果中可以看到，在文档内部定义的 DTD 在运行中使用“<!DOCTYPE student(*View Source for full doctype...*)>”指明文档类型。

2. 外部 DTD

外部 DTD 又称为 DTD 的外部子集，是指在 XML 文档外单独定义 DTD，并通过一个外部的 URL 将包含 DTD 的文件链接到 XML 文档中。外部 DTD 在物理位置上存在于另一个文件中，其扩展名为.dtd。由于外部 DTD 的独立性，所以可以供多个 XML 文档使用，当然 DTD 的定义结构必须与文档中的元素对应。

外部 DTD 可以使用两种格式声明，其语法分别如下所示：

```
<!DOCTYPE 根元素名 SYSTEM "文件名">
或
<!DOCTYPE 根元素名 PUBLIC "文件名">
```

这两种声明，都是以“<!DOCTYPE”关键字定义 DTD 文档类型，唯一的区别就是 SYSTEM 关键字和 PUBLIC 关键字。SYSTEM 表示该外部 DTD 文件是私有的，即由用户创建但并没有公开发行，只是在个人或几个合作者之间使用。PUBLIC 表示指该外部 DTD 文件是公有的，使用 PUBLIC 创建的 DTD 文件都有一个逻辑名称 DTD-name，必须在调用时指明这个逻辑名称。使用 PUBLIC 关键字通常表示 DTD 的使用范围比较大（例如 HTML 文档中也可以使用 DTD）。

下面通过示例说明如何定义一个外部 DTD，并且在 XML 文档中使用定义好的 DTD 文件。首先声明一个外部 DTD，如代码 12.7 所示。

代码 12.7　外部 DTD.dtd

```
<?xml version="1.0" encoding="GB2312"?>
<!ELEMENT student (name,id,class,body)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT class (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

将上述代码保存为“外部 DTD.dtd”。从代码中可以看出，外部 DTD 只是将文档类型声明中的内部 DTD 提取出来，保存为文件的形式，然后在 XML 文档中引用。接下来再创建一个 XML 文档，如代码 12.8 所示。

代码 12.8　外部 DTD.xml

```
<?xml version="1.0" encoding="GB2312" standalone="no"?>
<!DOCTYPE student SYSTEM "外部 DTD.dtd">
<student>
    <name>bob</name>
    <id>00102</id>
    <class>一年级一班</class>
    <body>Don't forget the meeting this weekend!</body>
</student>
```

在文档中引用外部 DTD 需要在 XML 声明中设置属性 standalone 的值为 no。如果外部 DTD 文件和 XML 文档存放于同一目录下，则可以直接使用 DTD 文件名；否则需要指明 DTD 文件所在的路径，该路径可以是绝对路径也可以是相对路径。在浏览器中运行 XML 文档，运行结果与图 12-2 相同。

12.2.3　声明 DTD

DTD 中主要包含 3 种类型声明，即元素类型声明、属性列表声明和实体声明。一个文档类型定义文件需要将创建的 XML 文档的元素结构和属性类型等预先进行定义。

1．元素类型声明

元素是 XML 文档中必不可少的一部分，用于显示 XML 文档的结构及特性。元素类型声明的语法格式如下：

```
<!ELEMENT 元素名 EMPTY /ANY/ #PCDATA /(子元素内容)/(混合内容)>
```

在该语法中，元素名后是元素内容类型。元素内容类型指定元素包含的数据内容的格式，主要分为 5 种类型：EMPTY（空）、ANY（任意）、#PCDATA、子元素类型和混合类型。

□ **EMPTY**

关键字 EMPTY 用于声明空元素，空元素只能包含属性，而不能有数据内容或子元素。声明空元素的语法格式如下：

```
<!ELEMENT 元素名 EMPTY>
```

例如声明一个空元素 name，语句如下所示：

```
<!ELEMENT name EMPTY>
```

XML 文档中对应声明的语句如下所示：

```
<name/> 或 <name></name>
```

□ **ANY**

使用 ANY 定义元素的类型，表示该元素可以是 DTD 中定义的其他任何元素或已编译的字符数据，包括#PCDATA、元素名或元素名和#PCDATA 的混合内容，而且还可以声明元素为空元素。ANY 类型的语法格式如下：

```
<!ELEMENT 元素名 ANY>
```

□ **#PCDATA**

#PCDATA 是被解析的字符数据。定义为#PCDATA 类型的元素不能包含任何子元素，而只能包含除标记外的任何字符数据，比如数字、字母和符号等。#PCDATA 类型的语法格式如下：

```
<!ELEMENT 元素名 (#PCDATA)>
```

例如：

```
<!ELEMENT name (#PCDATA)>
```

该语句表示元素 name 中不能包含子元素，而只能包含字符数据。如果包含有其他不属于#PCDATA 类型支持的内容，那么运行 XML 文档时就会出错。

□ **子元素类型**

元素可以包含其他子元素形成结构，而子元素类型定义了这种结构。子元素类型用于指定某个元素可以包含哪些子元素，其语法格式如下：

```
<!ELEMENT 元素名 (子元素名)>
```

在每个 XML 文档的 DTD 定义中必须为根元素定义子元素类型，根元素下包含的所有子元素都要使用该语句声明。例如：

```
<!ELEMENT student (name,id,class)>
```



在子元素类型的元素类型声明中，序列和选择两种结构可以同时使用。

□ **混合类型**

混合类型包含混合内容的元素，即一个元素既包含子元素又包含可解析的数据字符。在 DTD 中有 3 个可以使用的元素指示符，用于指定元素在文档中可出现的次数。加号“+”表示元素可以出现任意多次，但必须至少出现一次；星号“*”表示元素可以出现任意多次；问号“?”表示元素只能出现一次或不出现，如代码 12.9 所示。

代码 12.9　DTD 混合类型

```
<?xml version="1.0" encoding="GB2312" ?>
<!DOCTYPE student
  [<!ELEMENT student (name*)>
   <!ELEMENT name (#PCDATA)>
 ]>
<student>
   <name>叶简凝</name>
   <name>楚江南</name>
</student>
```

在该文档的 DTD 中定义根元素<student>和一个子元素<name>。在元素组中<name>可以出现任意多次。

2. 属性列表声明

属性是用于描述元素的额外信息。在 DTD 中使用 ATTLIST 关键字来定义元素的属性。属性声明要放在所在元素声明之后，其语法格式如下所示：

```
<!ATTLIST 元素名 属性名 属性类型 默认值>
```

一个元素的属性可以有零个或多个，并且每个属性的声明包括属性默认值、属性类型以及属性名称等。默认值表示在 XML 文件中元素属性的取值，以及属性是否为必须，其具有的默认值和说明如表 12-2 所示。

表 12-2　属性的默认说明

默认值	说明
#required	表示必须为元素提供此属性
#implied	表示标记中可以包含该属性，也可以不包含该属性
#fix	属性值是固定的某个值
字符串	标记中如果未指定属性值，则此字符串就是属性值

在 DTD 中属性类型有 10 种可供选择，表 12-3 列出了这 10 种属性类型及其意义。

表 12-3　属性类型及意义

属性类型	说明
CDATA	表示属性的取值为字符数据，与元素内容说明中的#PCDATA 相同
ENUMERATED	声明属性的取值为枚举类型
ID	ID 用属性值的方式为文件中某个元素定义唯一标识
IDREF	IDREF 类型允许一个元素的属性使用文件中另一个元素

续表

属性类型	说明
IDREFS	该类型的属性可以引用文档中多个 ID 类型的属性值。IDREFS 类型的属性值是一系列由空格分隔的 ID 类型的属性值
ENTITY	该类型的属性将外部的二进制数据链接到文档
NMTOKEN	名称符号，该类型的属性值是一种受限制的字符串
ENTITIES	ENTITIES 类型和 IDREFS 类型的使用是类似的，取值是由多个空格分隔的 ENTITY 类型的属性值
NMTOKENS	NMTOKENS 类型与 IDREFS 和 ENTITIES 类似，值由多个名称符号构成
NOTATION	NOTATION 类型的属性包含 DTD 中声明的某个记号的名称

3. 实体声明

在 DTD 中还可以声明一些实体（Entity），以便在 DTD 和 XML 文件中使用。实体属于类型的一种，可以将实体看作一个常量，其具有一定的值。在 DTD 中实体 Entity 的声明语法为：

```
<!ENTITY 实体名称 "实体的值">
```

例如，DTD 中可以采用如下形式声明实体：

```
<!ENTITY writer "Bill Gates">
```

实体声明好以后，在 XML 中使用的方法如下所示：

```
<author>&writer;</author>
```

引用实体时必须在实例名称前面加上“&”符号，后面加上“;”符号。实体可以分为内部实体、外部实体和参数实体。

内部实体是对文本字符串的引用，使用“&实体名;”就相当于插入引用的字符串。上面的示例就是一个内部实体的引用。外部实体是对其他外部文件的命名，它使得 XML 文件可以引用外部文件的内容。外部文件可以包含文本，也可以包含二进制数据。下面所示代码是一个关于外部实体的引用：

```
<!ENTITY entity_name SYSTEM "student.xml">
```

参数实体与前面两种实体不同，参数实体只能出现在 DTD 中。声明参数实体时，实例名称前面要加一个“%”和空格；引用参数实体时，引用其他实体时使用的“&”符号修改为“%”。下面所示代码是一个参数实体的声明：

```
<!ENTITY % student "学生 (姓名*)">
```

引用参数实体如下：

```
<!ELEMENT % student;>
```

12.2.4　DTD 实体

一个 XML 文档中的数据可以扩展分布在若干个文档中，其中包含 XML 文档细节内容的

存储单元称为实体。实体由一个文件、一个数据库记录或者其他包含数据的项目组成，例如一个完整的 XML 文件就是一个实体。包含 XML 声明或者文档类型声明的存储单元或基本元素称为文档实体。实体的主要目的在于保存结构完整的 XML、其他形式的文本或二进制数据。在 XML 中的 DTD 实体可以分为普通实体、参数实体和未解析实体。

1. 普通实体

普通实体是实体中应用比较广泛的一种实体。普通实体应用于 XML 文档的内容中，而且在 DTD 中可以使用普通实体插入文本，但是不可以在 DTD 中将其与它们的声明本身一起使用。根据文档对普通实体定义方式的不同，可以将其分为内部普通实体和外部普通实体两种。

- **内部普通实体**

内部普通实体引用可看作经常使用的文本或强制格式文本的缩写。DTD 中的<!ENTITY>标记定义缩写，并且使用该缩写代替文本。内部普通实体是指在 DTD 内部定义而被文档引用的实体。其语法格式如下：

```
<!ENTITY name "replacement text">
```

其中 name 是实体名称，用于替换 replacement text。替换文本需要放置在双引号中，因为其中可能包含空格和 XML 标记。

下面通过示例说明如何定义一个内部普通实体，并且在 XML 文档中引用。在文档中输入实体名称，则在页面中显示为替换文本，如代码 12.10 所示。

代码 12.10　内部普通实体

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<!DOCTYPE DOCUME[
 <!ENTITY mg "金融危机给我们带来了什么">
 <!ELEMENT DOCUME (title,body,content)>
 <!ELEMENT title (#PCDATA)>
 <!ELEMENT body (#PCDATA)>
 <!ELEMENT content (#PCDATA)>
]>
<DOCUME>
<title>&mg;</title>
<body>金融</body>
<content>
  金融危机又称金融风暴，是指一个国家或几个国家与地区的全部或大部分金融指标(如：短期利率、
  货币资产、证券、房地产、土地(价格)、商业破产数和金融机构倒闭数)的急剧、短暂和超周期的
  恶化。
</content>
</DOCUME>
```

在 DTD 中定义一个内部普通实体 mg，然后定义文档中的根元素及其子元素。运行该 XML 文档，文档中引用实体的地方都将以实体内容替代，如图 12-3 所示。



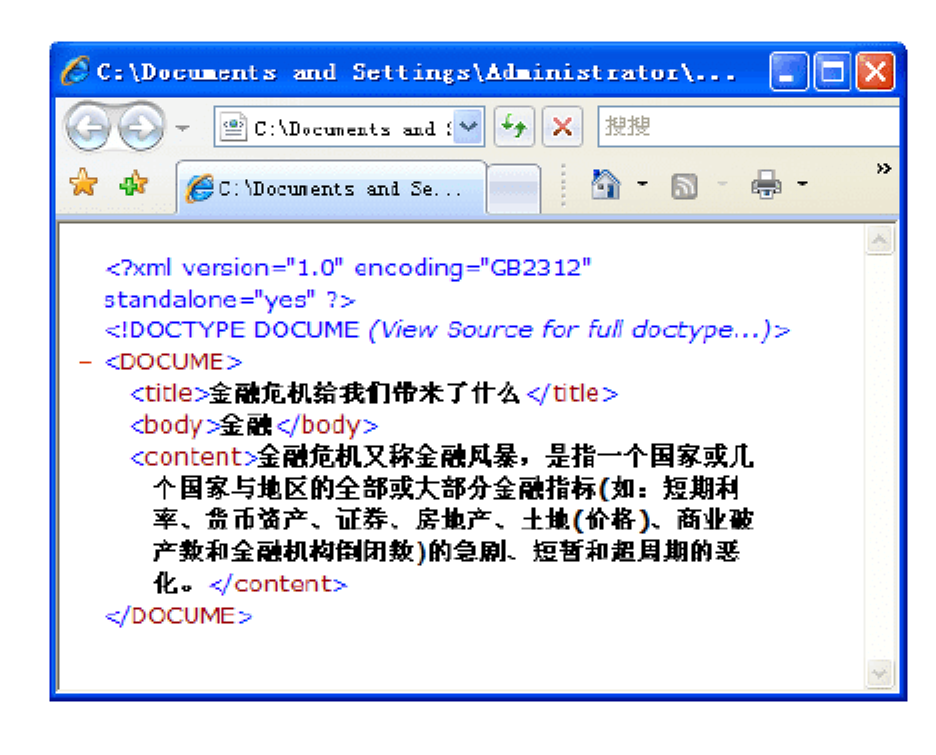


图 12-3　内部普通实体

□ 外部普通实体

外部普通实体是指实体引用的文件在文档外部，需要一个 URL 引用到 DTD 中才能被文档应用。其语法格式如下：

```
<!ENTITY name SYSTEM "URL">
```

下面通过一个示例说明如何引用外部普通实体。首先需要创建一个 XML 文档，命名为 conn.xml，在该文档中包含代码 12.11 所示信息。

代码 12.11　conn.xml

```
<?xml version="1.0" encoding="GB2312"?>
<Con>
<content>
金融危机又称金融风暴，是指一个国家或几个国家与地区的全部或大部分金融指标(如：短期利率、货币资产、证券、房地产、土地(价格)、商业破产数和金融机构倒闭数)的急剧、短暂和超周期的恶化。
</content>
</Con>
```

接下来再创建一个 XML 文档，并且在 DTD 中引用创建好的实体，具体实现如代码 12.12 所示。

代码 12.12　外部普通实体

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<!DOCTYPE DOCUME[
 <!ENTITY mg "金融危机给我们带来了什么">
 <!ENTITY content SYSTEM "conn.xml">
 <!ELEMENT DOCUME (title,body,content)>
 <!ELEMENT title (#PCDATA)>
 <!ELEMENT body (#PCDATA)>
 <!ELEMENT content (#PCDATA)>
]>
<DOCUME>
<title>&mg;</title>
```

```
<body>金融</body>
&content;
</DOCUME>
```

在 DTD 中，使用外部普通实体引用将一个 XML 文档的内容定义为一个实体，然后在 XML 文档中使用外部普通实体引用。运行该文档，结果与图 12-3 相同。

2. 参数实体

普通实体在 DTD 中定义，在与 DTD 关联的 XML 文档中引用，通过解析器最终将实体变为文档的一部分。而参数实体则只能在 DTD 中定义，并在 DTD 中使用，与关联的 XML 文档无关。参数实体引用的特点是以“%”开头，且只出现在 DTD 中。同样根据参数实体引用方式的不同，可以将其分为内部参数实体和外部参数实体两种。

内部参数实体是在 DTD 内部定义并使用的实体，其语法格式如下所示。其中 name 为参数实体的引用名称，text 为参数实体的内容。

```
<!ENTITY % name "text">
```

外部参数实体使用较小的 DTD 建立较大的 DTD，即一个 DTD 可以链接到另一个 DTD 上，从而得到该 DTD 中元素及属性的声明。外部参数实体的语法格式如下：

```
<!ENTITY % name "URL">
%name;
```

其中 name 为外部参数实体引用名称，URL 为外部 DTD 的存放路径。声明完成后外部参数实体还要通过“%name;”进行实体引用。

下面通过示例演示参数实体的使用。首先定义一个外部 DTD 用于表示图书的信息，如代码 12.13 所示。

代码 12.13　code.dtd

```
<?xml version="1.0" encoding="GB2312"?>
<!ELEMENT bookinfo (title,author,publish,price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

下面需要为图书添加一些出版社的信息，因此再添加一个名为 code1.dtd 的外部 DTD，如代码 12.14 所示。

代码 12.14　code1.dtd

```
<?xml version="1.0" encoding="GB2312"?>
<!ELEMENT publish (publisher,ISBN,pubdate)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT pubdate (#PCDATA)>
```

```
<!ENTITY % pub SYSTEM "code.dtd">
%pub;
```

该 DTD 中并没有给出图书信息元素的声明，而是定义一个外部参数实体 pub 引用 code.dtd 中定义的元素声明。接着使用 pub 的参数实体引用将元素声明引入。最后在 XML 文档中引入 code1.dtd 文件，如代码 12.15 所示。

代码 12.15　code.xml

```
<?xml version="1.0" encoding="GB2312" standalone="no"?>
<!DOCTYPE bookinfo SYSTEM "code1.dtd">
<bookinfo>
<title>计算机导论 </title>
<author>丁跃潮等 </author>
<publish>
<publisher>高等教育出版社 </publisher>
<ISBN>7-04-014768-8 </ISBN>
<pubdate>2004.6 </pubdate>
</publish>
<price>19.7 </price>
</bookinfo>
```

在浏览器中查看运行结果，如图 12-4 所示。

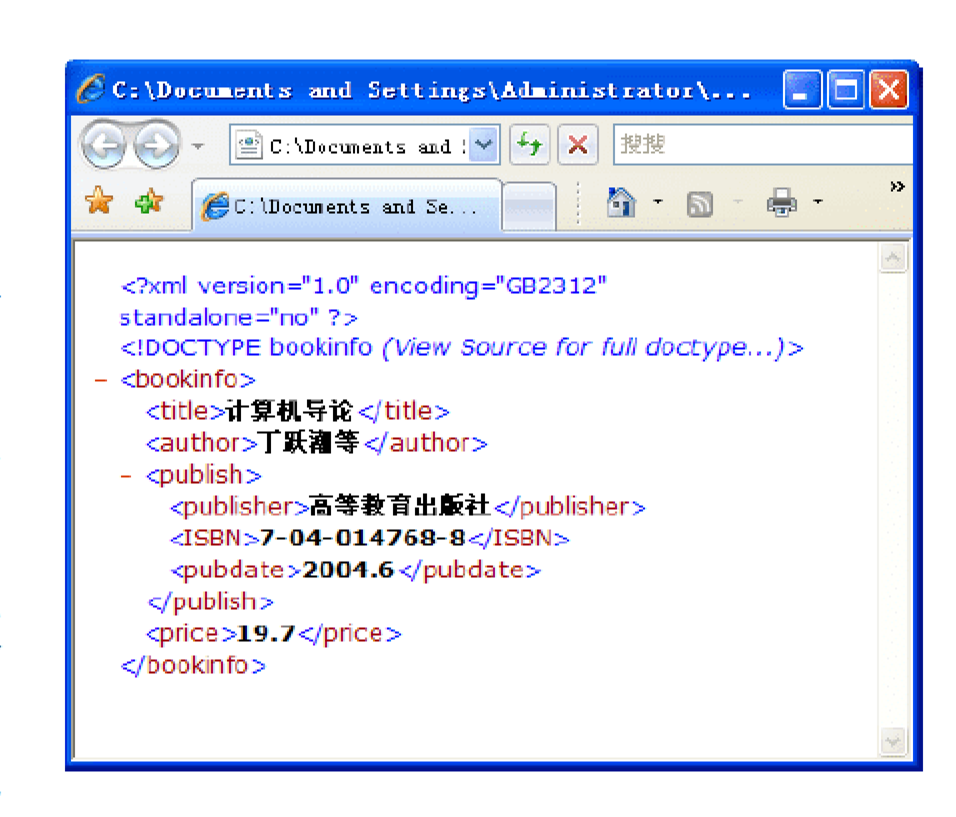


图 12-4　参数实体

3. 未解析实体

在 XML 中，实体还可以根据是否能被解析器解析将其分为可解析实体和未解析实体。可解析实体就是能够被解析器解析的实体，该实体一般由字符数据所构成；而未解析实体就是不能够被解析器解析的实体。例如在 XML 文档中引用图像、声音等非文本数据，这些数据是解析器所不能解析的，就是未解析实体。

未解析实体通常指二进制数据，XML 文档不能够直接引用。首先未解析实体提供非 XML 数据的真正位置的链接，然后使用 ENTITY 类型的属性链接到该实体和文档中的特定元素。未解析实体的语法格式有如下两种形式：

```
<!ENTITY name SYSTEM "URL">
```

或

```
<!ENTITY name SYSTEM "URL" NDATA type>
```

在这两种形式中，name 指定未解析实体的名称。第一种语法形式中 URL 链接的只能是形式规范的 XML 文档，例如：

```
<!ENTITY student SYSTEM "student.xml">
```

如果未解析实体引用的是声音、图像等二进制数据，而非 XML 文档，则可以使用第二种语法形式。该语法中 URL 链接的是所需要的文件，type 为该文件的类型，例如：

```
<!ENTITY picture SYSTEM "image.jpg" NDATA JPEG>
```


此例中声明未解析实体 picture 引用图像文件 image.jpg，而非 XML 文档，因而需要使用 NDATA 表示 XML 不解析该数据，并声明其类型。

12.3　XML 架构 Schema

XML Schema 作为一种描述 XML 文档模式信息即结构信息的标准，对于 XML 索引的建立及查询效率的提高有着重要的作用。现有的大部分 XML 索引结构着重研究 XML 文档的结构查询，而对于 XML 文档更新的支持却不多，对无效查询也不能做出快速的判断。

XML Schema 由预先定义的 XML 元素和属性创建，这些元素和属性定义文档的结构和内容模式。XML Schema 的特性如下所示。

- **一致性**　XML Schema 直接利用 XML 的基本语法规则定义文件结构。
- **扩展性**　XML Schema 对 DTD 进行扩充，引入数据类型和命名空间，使其具备更强的可扩展性，而且一个 Schema 文件中可以引用其他 Schema 文件或者在相同的文档中参考多种 Schema。
- **互换性**　XML Schema 可验证 XML 文件的合法性，并通过特定的映射机制将不同的 XML Schema 进行转换，实现高层次的数据交换。
- **规范性**　XML Schema 提供完整的机制以约束 XML 标记的使用，利用元素的内容和属性定义整体结构。
- **数据类型多样性**　XML Schema 支持丰富的数据类型，例如整型和布尔型等。

使用 XML Schema 的好处是它本身就是一个 XML 文档，只要掌握 XML 语法或规则就能够很快地掌握 XML Schema 的使用。使用 Schema 描述 XML 文档已成为 XML 发展的一种趋势。

12.3.1　XML Schema 模型结构

XML Schema 有两种重要的 Schema 模型，分别是 Microsoft XML Schema 和 W3C XML Schema。这两种模型定义的 XML 文档结构相同，只是在结构声明、描述、命名空间的使用和 Schema 文件扩展名有所不同。XML Schema 文档由元素、属性、命名空间和 XML 文档中的其他元素构成，其中至少要包含 Schema 根元素、XML 模式命名空间和元素的定义。在 W3C XML Schema 模型结构中必须以<xsd:schema>或<xs:schema>开始列为文档根元素，结构中所有的元素及属性声明均是以 xsd:或 xs:开头。xsd（或 xs）为 XML Schema 命名空间的别名，XML Schema 引用的命名空间为 http://www.w3.org/2001/XMLSchema。定义 XML Schema 文档的结构如下所示：

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
元素声明部分或属性声明部分
</xsd:schema>
```

12.3.2　XML Schema 数据类型

XML Schema 在描述 XML 文档的结构时，最突出的方面就是支持丰富的数据类型，可以是元素或属性包含内容的类型。XML Schema 中的数据类型可以分为简单类型和复杂类型两种。简单类型是不能分割的原子信息；复杂类型类似于编程语言中的用户自定义类型，是由已存在的简单类型组合而成。

1. 简单类型

XML Schema 规范定义两种简单类型，内置类型和用户自定义类型。内置类型又可分为基础类型和派生类型两类。基础类型是解析系统直接支持的原始类型，派生类型是对基础类型或其他的内置派生类型加以限制而生成的。用户自定义类型是对内置类型或其他用户自定义类型加以限制或扩展而生成的。

定义简单类型的元素只能包含数字、字符或其他文本数据，而不能包含子元素。简单类型元素声明以<xsd:simpleType>开始，属性 name 为文档根元素的名称。定义简单类型的语法结构为：

```
<xsd:simpleType name = 名称 id = id>
<annotation | restriction | list | union>…
</xsd:simpleType>
```

上述语法中的属性 name 表示类型名称，该名称必须是在 XML 命名空间规范中定义的无冒号名称。属性 id 表示元素的 id，id 值必须属于类型 id 并且在包含该元素的文档中是唯一的。包含内容列表可以在简单类型中定义的元素类型。表 12-4 对这些类型做详细说明。

表 12-4　简单类型元素

元素	说明
annotation	批注定义，用于对 XML Schema 文档中某一部分进行说明。批注部分不属于 XML 文档中的内容
restriction	为简单类型、简单内容或复合内容定义一个限制属性，即限制基类型
list	定义简单类型的列表类型
union	定义简单类型的联合类型

简单类型分为原子类型、列表类型和联合类型 3 种，这 3 种类型综合起来称为简单类型。下面分别介绍这几种类型。

□ **原子类型**

原子类型的值具有不可分割性，一般来说 XML Schema 内置的数据类型就属于原子类型。表 12-5 列出了 XML Schema 中几种常用的内置数据类型。

表 12-5　常用的内置数据类型

数据类型	说明
string	字符串型
integer	整型
decimal	十进制数型，包含任意精度和位数
float	标准的 32 位浮点数，例如 11.87
boolean	布尔型，元素只能取真或假的值
date	天日期型，格式为 YYYY-MM-DD
month	月份日期型，格式为 YYYY-MM
year	年份日期型，格式为 YYYY
time	时间型，格式为 HH:MM:SS
datetime	日期时间型，其形式为 YYYY-MM-DD hh:mm:ss

下面通过示例演示使用原子类型的 Schema，并且在 XML 文档中引用，通过<xsd:restriction>的 base 属性指定两种数据类型，具体实现如代码 12.16 所示。

代码 12.16　原子类型.xsd

```
<?xml version="1.0" encoding="GB2312"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:simpleType name="book">
<xsd:restriction base="xsd:string">
  <xsd:enumeration value="Title" />
  <xsd:enumeration value="Author" />
</xsd:restriction>
<xsd:restriction base="xsd:integer">
   <xsd:enumeration value="Page" />
   <xsd:enumeration value="Price" />
</xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

代码中使用<xsd:simpleType>声明一个包含简单类型元素的文档。<xsd:restriction>是一个限制基类型元素，可以限制现有的简单类型的值空间以满足编写文档的需要。其属性 base 指定了两种数据类型 string 和 integer。string 类型说明只能包含字符，integer 类型说明只能包含整数。

□　**列表和联合类型**

列表类型是其值使用空格隔开的原子值类表。列表类型使用<xsd:list>声明，属性 itemType 指定元素数据类型，并声明了 XML 文档中的元素引用该列表类型。

联合类型的值可以是原子值，也可以是列表值。联合类型可以包含多个原子类型或者列表类型。它使用<xsd:union>声明元素，使用 memberType 定义要包含的类型值。

下面通过示例演示如何引用列表类型。首先创建一个 Schema 文件，在该文件中引用列表类型，具体实现如代码 12.17 所示。

代码 12.17　List.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:simpleType name="book">
<xsd:restriction base="xsd:string">
     <xsd:enumeration value="c#"/>
     <xsd:enumeration value="Java"/>
     <xsd:enumeration value="Struts"/>
     <xsd:enumeration value="JSF"/>
     <xsd:enumeration value="ASP.NET"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="BookList">
<xsd:list itemType="book" />
</xsd:simpleType>
<!--输出 3 个图书信息-->
<xsd:simpleType name="fristBook">
<xsd:restriction base="book">
<xsd:length value="3" />
</xsd:restriction>
</xsd:simpleType>
<xsd:element name="BookInfo" type="BookList">
<xsd:element name="asd" type="fristBook">
</xsd:schema>
```

Schema 文件创建好以后就可以在 XML 文件中引用，在 XML 文档中应该与列表类型中的元素相一致，如代码 12.18 所示。

代码 12.18　List.xml

```
<?xml version="1.0"?>
<catalog xmlns:xsd="http:www.w3.org/2001/XMLSchema"
  xsd:noNamespaceSchemaLocation="List.xsd">
<BookInfo>
c# Java Struts JSF ASP.NET
</BookInfo>
<asd>
c# Java Struts
</asd>
</catalog>
```

2. 复杂类型

复杂类型的元素有子元素和属性，也可以有字符内容。在 XML Schema 中复杂类型由<complexType>元素定义，至少有一个 name 属性用在声明其他元素时索引类型，除非它位于

某一元素之内。复杂类型的语法格式如下所示：

```
<complexType name = 名称 id = ID mixed =true|false >
<annotation|simpleContent|complexContent|group|all|choice|sequence|
attribute
|attributeGroup|anyAttribute>…
</complexType>
```

语法中各参数含义如下所示。

- **name**　复杂类型的名称，该名称必须为在 XML 命名空间中规范定义的无冒号名称。
- **id**　该元素的 id，id 值必须属于类型 id 并且在包含该元素的文档中是唯一的。可选项。
- **mixed**　一个指示符，指示是否允许字符数据出现在复杂类型的子元素之间。默认值为 false。如果 simpleContent 元素是子元素，则不允许重写 mixed 属性；如果 complexContent 元素是子元素，则 mixed 属性可被 complexContent 元素的 mixed 属性重写。可选项。
- **包含内容列表**　其中的元素内容均可以被定义在复杂类型中，含义如表 12-6 所示。

表 12-6　复杂类型元素

元素	说明
annotation	批注，只在 XML Schema 中起注释的作用，并不影响架构内容
simpleContent	从简单类型派生复杂类型。适用于包含字符内容和属性但不包含子元素的元素声明
group	将若干元素声明归为一组，以便将其当作一个组并入复杂类型定义
all	表示符合元素声明的所有元素都应该出现（以任何顺序）且只能出现一次
choice	允许使用通过<choice>声明的嵌套元素中的一个元素
sequence	定义一系列元素并且必须按照模式中指定的顺序显示，所有子元素如果在实例文档中出现的话，都必须按照该顺序出现
attribute	为出现的元素定义属性
attributeGroup	定义属性列表
anyAttribute	在 XML 文档中添加未被 schema 指定过的属性
complexContent	从一个复杂类型派生出新的复杂类型。适用于包含属性和子元素但不包含字符数据（字符数据包含在子元素中）的元素声明

所有复杂类型都会包含一个内容定义类型，其主要功能是定义类型所能包含的内容模型。复杂类型子元素的顺序和结构称为内容模板。下面通过示例说明如何创建一个复杂类型的 Schema，如代码 12.19 所示。

代码 12.19　complex.xsd

```
<?xml version="1.0" encoding="GB2312"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="student">
 <xsd:complexType>
   <xsd:sequence>
     <xsd:element name="name" type="xsd:string" />
     <xsd:element name="sex" type="xsd:string" />
     <xsd:element name="grade" type="xsd:string" />
```

```
        <xsd:element name="number" type="xsd:integer" />
        <xsd:element name="other">
          <xsd:complexType>
          <xsd:element name="brithday" type="xsd:date" />
          <xsd:element name="address" type="xsd:string" />
          </xsd:complexType>
        <xsd:element>
      </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   </xsd:schema>
```

Schema 文件创建好以后，接下来创建与 Schema 对应的 XML 文档，如代码 12.20 所示。

代码 12.20　complex.xml

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<student xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xsd:noNamespaceSchemaLocation="complex.xsd">
<name>Anne Lee</name>
<sex>女</sex>
<grade>一年级二班</grade>
<number>1259000022</number>
<other>
<brithday>1988-01-25</brithday>
<address>London</address>
</other>
</student>
```

12.3.3　XML Schema 元素声明

XML Schema 中的元素使用 element 标识符声明。name 属性是元素的名字；type 属性是元素值的类型，可以是 XML Schema 中内置的数据类型或其他类型。元素声明的语法如下所示：

```
<xsd:element abstract|block|default|final|fixed|form|id|maxOccurs|minOccurs
|name|
nillable|ref|substitutionGroup|type>
<xsd:annotation|xsd:simpleType|xsd:complexType|xsd:unique|xsd:key|xsd:keyr
ef>…
</xsd:element>
```

对于元素 element 的全部属性及子元素说明如表 12-7 和表 12-8 所示。

表 12-7　element 的属性

属性名	说明
abstract	一个指示符，boolean 型。指示元素是否可以在实例文档中使用。如果该值为 true，则元素不能出现在实例文档中。默认值为 false
block	派生的类型。block 属性防止具有指定派生类型的元素被用于替代该元素
default	如果元素的内容是简单类型或者是 textOnly，则为元素的默认值
final	派生的类型。final 属性在 element 元素上设置 final 属性的默认值
fixed	如果元素的内容是简单类型或者是 textOnly，则为该元素预定的、不可更改的值
form	元素的形式。默认值是包含该属性的<schema>元素的 elementFormDefault 属性的值，该值必须是 qualified 或 unqualified。如果该值是非限定的，则无须通过命名空间前缀限定该元素。如果该值是限定的，则必须通过命名空间前缀限定该元素
id	元素的 id。id 值必须属于类型 id 并且在包含该元素的文档中是唯一的
maxOccurs	元素可以在包含元素中出现的最大次数。该值可以是大于或等于零的整数
minOccurs	元素可以在包含元素中出现的最小次数。该值可以是大于或等于零的整数
name	元素的名称
nillable	一个指示符，boolean 型。指示是否可以将显式的零值分配给该元素。应用于元素内容并且不是该元素的属性。默认值为 false
ref	对另一个元素的引用。ref 属性可包含一个命名空间前缀。如果父元素是<schema>元素，则不是使用该属性
substitutionGroup	用于替代该元素的名称。该元素必须具有相同的类型或从指定元素类型派生的类型
type	指定元素数据类型，该数据类型可以是内置数据类型，或者是在此架构（或者由指定命名空间指示的其他架构）中定义的<simpleType>或<complexType>元素。提供的值必须与引用的<simpleType>或<complexType>元素上的<name>属性值相对应

表 12-8　element 的子元素

元素名	说明
annotation	定义批注
simpleType	定义一个简单类型，确定与具有纯文本内容的属性或元素的值有关的信息以及约束
complexType	定义一个复杂类型，确定属性集和元素内容
unique	指定属性或元素值（或者元素值的组合）在指定范围内必须是唯一的。该值必须唯一或为零
key	指定属性或元素值（或一组值）必须是指定范围内的键。键的范围为实例文档中的包含 element。键意味着数据在指定范围内应是唯一的、不为零的并且始终存在
keyref	指定属性或元素值（或一组值）与指定的<key>或<unique>元素的值相对应

在 XML Schema 中，对于元素的定义可以有选择地使用上述介绍的属性及子元素。下面创建一个 XML 文档，然后根据该文档编写 XML Schema 架构。XML 文档的内容如代码 12.21 所示。

代码 12.21　Element.xml

```
<?xml version="1.0" encoding="GB2312" standalone="yes" ?>
<Student>
<student grade="一年级一班">
 <name>bob</name>
 <number>00102</number>
```

```
</student>
<student grade="一年级二班">
 <name>Anne</name>
 <number>1295000022</number>
</student>
<student grade="一年级三班">
 <name>Poe</name>
 <number>1295000301</number>
</student>
</Student>
```

在 XML 文档中根元素<Student>中有一个子元素<student>，但是元素及属性的内容各不相同。元素<student>中包含子元素<name>和<number>，以及属性 grade，对应的 XML Schema 架构如代码 12.22 所示。

代码 12.22　Element.xsd

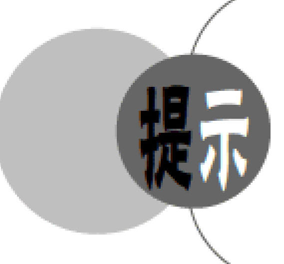
```
<?xml version="1.0" encoding="GB2312"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="Student">
<xsd:complexType>
<xsd:sequence>
 <xsd:element name="student" minOccurs="1" maxOccurs="unbounded">
   <xsd:compleType>
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs=
        "2" />
        <xsd:element name="number" type="xsd:integer">
      </xsd:sequence>
    <xsd:attribute name="grade" type="xsd:string" use="required">
   </xsd:compleType>
 </xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

在上述代码中，首先使用<xsd:element>元素定义根元素<Student>，由于包含有子元素及属性，所以声明为复杂类型，并让元素按语句中的顺序在文档中出现。然后定义子元素<student>，及其子元素<name>和<number>，还有属性 grade。其中元素<student>和<name>的出现次数至少一次。grade 指定 use 属性为 required，表示属性 grade 在元素<student>中必须出现。

当架构生成后在 XML 中引用该架构，并运行文档进行有效型校验。引用代码如下所示：

```
<Student xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsd:noNamespaceSchemaLocation="Element.xsd">
```

```
    </Student>
```



在 XML Schema 架构中必须定义一个且只能定义一个 Schema 根元素。根元素中包括模式的约束、XML 模式命名空间的定义，以及其他命名空间的定义、版本信息、语言信息和其他一些属性。

12.3.4　XML Schema 属性声明

XML Schema 属性声明由<xsd:attribute>元素来实现，并且可以直接在元素声明内定义。<xsd:attribute>元素的语法格式如下：

```
    <xsd:attribute default|fixed|form|id|name|ref |type |use>
    <xsd:annotation|xsd:simpleType>…
    </xsd:attribute>
```

<xsd:attribute>元素的各属性及子元素如表 12-9 和表 12-10 所示。

表 12-9　attribute 的属性

属性名	说明
default	指定属性的默认值
fixed	指定属性的固定值
form	指定属性的格式。默认值是包含该属性的<schema>元素的 attributeFormDefault 属性的值。该值必须是字符串 qualified 或 unqualified
id	该元素的 id。id 值必须属于类型 id 并且在包含该元素的文档中是唯一的
name	指定属性的名称
ref	在该架构（或由指定的命名空间指示的其他架构）中声明的属性的名称。ref 值必须是限定名。类型可以包括命名空间前缀。name 和 ref 属性不能同时出现。如果 ref 出现，则<simpleType>元素、form 和 type 不能出现
type	指定属性的数据类型
use	指示如何使用属性。有效值是 optional（该属性是可选的并且可以具有任何值）、prohibited（该属性用于其他复杂类型的限制中以禁止使用现有属性）和 required（该属性是必需的并且可以包含该属性的此类型定义允许的任何值）

表 12-10　attribute 的子元素

元素名	说明
annotation	定义批注
simpleType	定义一个简单类型

在复杂类型的语法定义中有 3 个与属性相关的元素，分别是<attribute>、<attributeGroup>和<anyAttribute>。<attributeGroup>用于定义一个属性列表，<anyAttribute>可以让用户在 XML 文档中任意扩展属性，即可以使用任意属性而不用在 XML Schema 中定义。

在前面的示例中简单的使用了属性的定义，下面创建一个 XML 文档，然后结合属性声明中的元素及属性编写 XML Schema 架构。XML 文档的内容如代码 12.23 所示。

代码 12.23　attribute.xml

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<student name="Anne" sex="女" number="1295000022">
   <grade id="一年级二班">班级</grade>
   <school>第十八中学</school>
</student>
```

XML 文档创建好以后，接下来创建 XML Schema 架构，如代码 12.24 所示。

代码 12.24　attribute.xsd

```
<?xml version="1.0" encoding="GB2312"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="Student">
  <xsd:complexType>
    <xsd:sequence>
       <xsd:element name="grade">
         <xsd:complexType>
           <xsd:simplecontent>
             <xsd:extension base="xsd:string">
               <xsd:attribute name="id" type="xsd:string" use="required"/>
             </xsd:extension>
           </xsd:simpleContent>
         </xsd:complexType>
       </xsd:element>
       <xsd:element name="school" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="student">
  </xsd:complexType>
</xsd:element>
<xsd:attributeGroup name="student">
<xsd:attribute name="name" default="Anne"/>
<xsd:attribute name="sex" default="女"/>
<xsd:attribute name="number" default="1295000022"/>
</xsd:attributeGroup>
</xsd:schema>
```

在该代码中使用<attributeGroup>元素定义属性组 student。一个元素的多个属性声明可以定义在一个属性组中，然后再通过 ref 元素将属性组连接到所属元素的属性声明中，最后将 XML Schema 架构引用到 XML 文档中。

12.4　XSLT

XSLT 是依据 XML 指定的样式语言，通过一系列的语法规则使 XML 文档得到更加有效

的表现形式，从而实现文档内容和表现形式的分离。使用 XSLT 可以使 XML 像 HTML 那样显示数据。

12.4.1　XSLT 简介

XSLT 是 eXtensible Stylesheet Language Transformation 的英文缩写，即可扩展的样式表转换。在介绍 XSLT 之前，先来介绍一下 XSL（eXtensible Stylesheet Language，可扩展的样式表语言）。XSL 本来用于两个目的：XML 数据表达和数据转换。逐渐地数据转换部分从中独立出来形成一个新的技术即 XSLT，而数据表达部分则形成 XSL-FO。XSLT 是 XSL 标准中最重要的一个部分。

XSLT 语言由 W3C 定义，并且在 1999 年 11 月 16 日将该语言的 1.0 版本作为“推荐书”发布。最初设计 XSLT 的目的是用于帮助 XML 文档转换成其他文档。但随着发展，XSLT 已经成为用于转换 XML 文档结构的语言。

XSLT 是为 XML 样式显示设计的语言，该语言定义了一系列元素集的 XML 语法规范。该语法规定可将 XML 文档转换成 HTML 或其他的文档，也就是说将一个 XML 文档转换成浏览器所能识别的一种格式。一个 XSLT 的样式表单里包含多个设计规则、显示方式，从 XML 文档中提出来的数据依据这个样式表单规定的显示方式显示，形成一个新的文档。

XSLT 最常用的转换是将 XML 文档转换成 HTML 文档。XSLT 将每个 XML 元素都转换成一个 HTML 元素。XSLT 还可以向输出文件中增加新的元素或者去掉一些元素，可以重新安排这些元素并对元素进行分类，测试并确定显示哪些元素等。

相对于 CSS 只支持英文的限制，XSLT 最大的优点就是文档中可以支持中文。这样可以更加便利于用户的编写习惯。而且目前的浏览器版本具备了对 XSLT 文档支持的带有 XSLT 引擎的 XML 解析器，所以用户可以直接使用 Internet Explorer 5.0 或者以上版本来调试 XSLT 文档。

12.4.2　XSLT 文档

XSLT 可以将 XML 转换成 HTML 形式、WAP 形式输出，甚至还可以是 Word 文档的格式。其中最常用的是 HTML 的格式。下面首先创建一个 XML 文档，然后再创建一个 XSLT 文档，使用 XSLT 文档转换 XML 文档并以 HTML 格式显示 XML 文档中的内容，XML 文档的内容如代码 12.25 所示。

代码 12.25　Employee.xml

```
<?xml version = "1.0" encoding="GB2312"?>
<resumes>
<resume>
<name>小小</name>
<sex>女</sex>
<age>24</age>
<birthday>
```

```
<year>1986</year>
<month>8</month>
<day>10</day>
</birthday>
<address>北京市 海淀区</address>
</resume>
<resume>
<name>江南</name>
<sex>男</sex>
<age>27</age>
<birthday>
<year>1982</year>
<month>6</month>
<day>9</day>
</birthday>
<address>山西省</address>
</resume>
</resumes>
```

XSLT 处理器在解读 XSLT 与 XML 文件后便可以产生相对应的输出结果，而这个结果可以被浏览器所显示。由于 XSLT 设置的样式只用于 XML 文档，所以 XSLT 文件也必须是格式规范且有效。XSLT 文件的扩展名为.xslt。根据 Employee.xml 创建 XSLT 文件 Employee.xslt，文件语句如代码 12.26 所示。

代码 12.26　Employee.xslt

```
<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
<head>
<title>个人简历</title>
</head>
<body>
<xsl:for-each select="/resumes/resume">
<p>
<table border="1" >
<caption style="font-size: 150%; font-weight: bold">
 个人简历
</caption>
<tr>
<th>姓名</th><td><xsl:value-of select="name"/></td>
<th>性别</th><td><xsl:value-of select="sex"/></td>
<th>生日</th><td><xsl:value-of select="birthday/year"/>年
<xsl:value-of  select="birthday/month"/> 月 <xsl:value-of  select="birthday/
day"/>日</td>
```

```
</tr>
<tr>
<th>地址</th><td colspan="5"><xsl:value-of select="address"/></td>
</tr>
</table>
</p>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

将创建好的 XSLT 文件引用到 XML 文档中，需要在 XML 声明语句下使用下列引用语句：

```
<?xml:stylesheet href=" Employee.xslt " type="text/xsl" ?>
```

在上述语句中，xml:stylesheet 表示将 XSLT 样式文件应用到 XML 文档上。属性 href 表示要引用的 XSLT 文件的名称，如果 XSLT 文件与所应用的 XML 文档在同一目录下则可以直接写入名称；如果不在则可以指定具体路径。还可以使用 URL 提供有效的链接地址来引用 XSLT 文件。属性 type 定义要引用的文件类型，引用 XSLT 文件时的值为"text/xsl"。

最后运行 Employee.xml 文档，运行结果如图 12-5 所示。

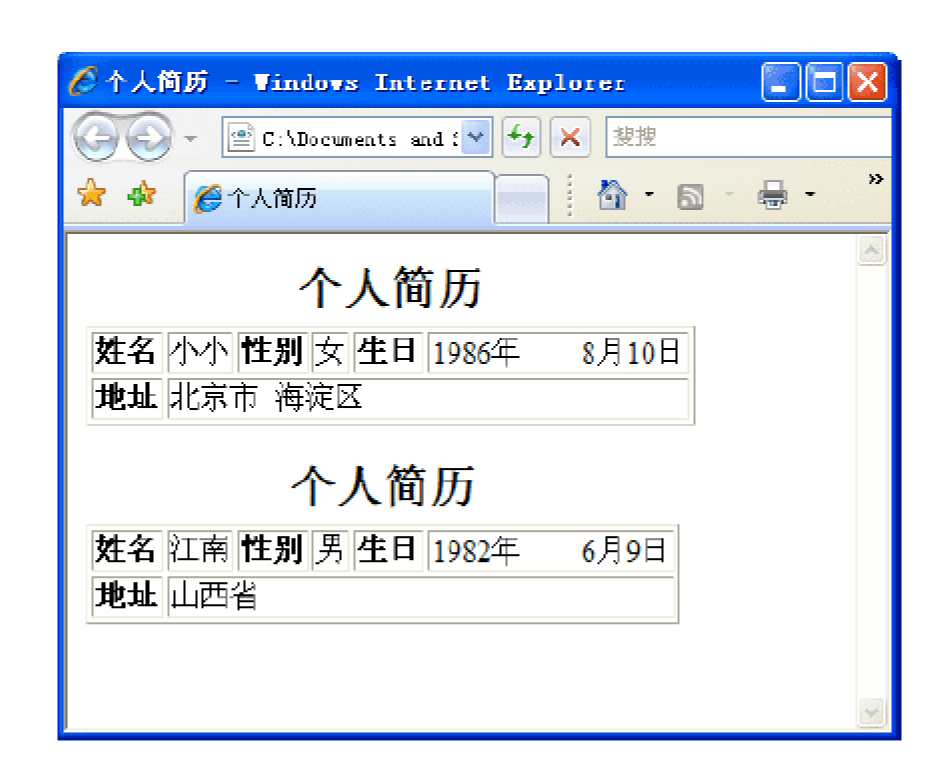


图 12-5　使用 XSLT 文档

12.4.3　XSLT 模板语法

模板（template）是 XSLT 中最重要的概念之一。XSLT 文件就是由多个模板组成的，任何一个 XSLT 文件至少包含一个模板。模板由两部分组成：匹配模式（match pattern）和执行。模式定义 XML 源文档中哪一个节点将被模板处理，执行则定义输出的是什么格式。两部分对应的语法分别为 xsl:template 和 xsl:apply-templates。

- **xsl:template 语法**

```
<xsl:template match = pattern name = qname priority = number mode = qname>
<!--执行内容-->
</xsl:template>
```

xsl:template 的作用是定义一个新模板。name、priority 和 mode 属性用于区别匹配同一节点的不同模板，它们不是常用的属性。match 属性则控制模板的匹配模式（pattern），匹配模式是用于定义 XML 源文档中哪一个节点被模板处理，一个模板匹配一个节点。

match 是模板的必需属性，值必须为与 XML 文档标记相匹配的字符串，称为标记匹配模式。标记匹配模式用于指定要从 XML 文档中哪个标记处开始寻找和提取数据。如果没有为

match 设置属性值，表示从当前的根元素处开始查找和提取数据，其默认值为“/”。如果要设置属性值，则需要根据 XML 文档的树状结构给出要提取数据元素的完整路径。

假设要处理一个包含章节和段落的文档。使用<para>元素定义段落，<chapter>元素定义章节，下面查看 match 属性可能的值。下面的语句写法说明模板匹配所有的<para>元素：

```
<xsl:template match="para">
模板内容
</xsl:template>
```

下面的语句写法说明模板匹配所有的<para>元素和所有的<chapter>元素：

```
<xsl:template match="(chapter|para)">
模板内容
</xsl:template>
```

下面的语句写法说明模板匹配所有的父节点为<chapter>元素的<para>元素：

```
<xsl:template match="chapter//para">
模板内容
</xsl:template>
```

在一个 XSLT 样式表中必须存在一个根模板，根模板是与 XML 文档中根标记匹配的模板。但是如果使用浏览器处理 XSLT 转换，则根模板的标记匹配模式必须是“/”。下面的语句写法说明模板匹配根节点：

```
<xsl:template match="/">
模板内容
</xsl:template>
```

当处理器进行 XML 文档转换时必须先查找到根模板，然后再从根模板开始进行 XSLT 转换。根模板的内容中可以包括其他模板调用的标记，如果样式表中没有根模板，那么该样式表就不起任何作用。

- **xsl:apply-templates 语法**

```
<xsl:apply-templates select = node set-expression mode = qname>
模板内容
</xsl:apply-templates>
```

<xsl:apply-templates>用于定义执行时哪一个节点被模板具体处理，可以理解为程序中调用子函数。select 属性用于定义确切的节点名称。<xsl:apply-templates>总是包含在<xsl:template>元素中，如下所示：

```
<xsl:template match="/">
<xsl:apply-templates select="para"/>
</xsl:template>
```

这段代码说明模板匹配整个文档（根节点），具体执行时处理根节点下所有<para>元素。下面通过示例说明如何根据 XML 文档定义模板的使用方法。XML 文档的内容如代码 11.27

所示。

代码 11.27 template.xml

```
<?xml version="1.0" encoding=" GB2312"?>
<Student>
 <name>Anne Lee</name>
 <sex>女</sex>
 <brithday>1988-01-25</brithday>
 <address>London</address>
</Student>
```

创建 XSLT 文件如代码 12.28 所示。

代码 12.28 template.xslt

```
<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html>
   <body>
     <xsl:apply-templates select ="Student/name"/>
     <xsl:apply-templates select ="Student/sex"/>
     <xsl:apply-templates select ="Student/brithday"/>
     <xsl:apply-templates select ="Student/address"/>
   </body>
  </html>
</xsl:template>
<xsl:template match="Student/name">
<p>name:
<xsl:value-of select="." /></p>
</xsl:template>
<xsl:template match="Student/sex">
<p>sex:
<xsl:value-of select="." /></p>
</xsl:template>
<xsl:template match="Student/brithday">
<p>brithday:
<xsl:value-of select="." /></p>
</xsl:template>
<xsl:template match="Student/address">
<p>address:
<xsl:value-of select="." /></p>
</xsl:template>
</xsl:stylesheet>
```

在 XSLT 文件中，首先是 XML 和 XSLT 声明，然后声明根模板。在根模板中，XSLT 处

理器对 XML 文档进行转换。XSLT 处理器首先从根模板处进行转换，如果发现根模板存在模板调用元素<xsl:apply-templates>，就会根据这里的标记匹配模式到 XML 文件中寻找所有和标记匹配模式相匹配的标记，当找到这些标记后，再到 XSLT 文件中为这些标记寻找相应的模板，一旦找到该标记匹配的模板就会对该标记的内容实施 XSLT 转换，并将转换后的文本嵌入到 HTML 中。

在 XML 文档中引用 XSLT 文件：

```
<?xml:stylesheet href="template.xslt" type="text/xsl"?>
```

运行 XML 文档，运行结果如图 12-6 所示。

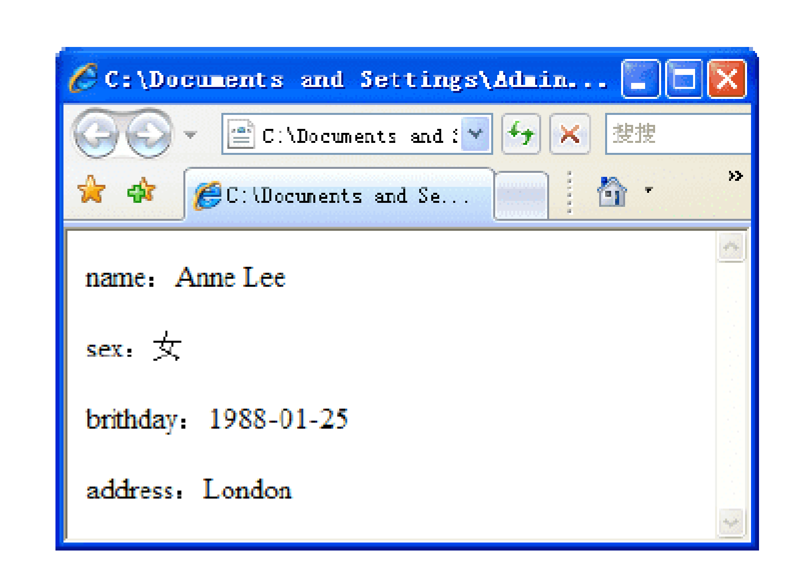


图 12-6 调用模板

12.4.4 XSLT 元素

XSLT 样式表的基本结构就是模板，通过模板对 XML 文档进行相应的转换。通过前面的介绍，读者可能已经对转换过程有一定的了解。XSLT 样式表转换 XML 文档的过程离不开 XSLT 元素，各元素所起的作用是不同的，表 12-11 列出了 XSLT 元素及其作用。在本小节中将对 XSLT 样式表中几种重要元素的作用及用法进行详细的介绍。

表 12-11 XSLT 元素

元素	描述
xsl:apply-imports	使用一个导入的样式表来实现模板规则
xsl:apply-templates	将模板规则应用到当前元素或者当前子节点元素
xsl:attribute	添加一个属性
xsl:attribute-set	定义一个属性集的名字
xsl:call-template	调用一个命名模板
xsl:choose	与<when>和<otherwise>元素配合使用可用于表示多条件测试
xsl:comment	在结果树中创建一个备注节点
xsl:copy	创建一个当前节点的副本（子节点和属性不复制）
xsl:copy-of	创建一个当前节点的副本（同时复制子节点和属性）
xsl:decimal-format	定义转换数字为字符串时使用的字符和标志，需要使用 format-number()函数
xsl:element	在输出文档中创建一个元素节点
xsl:fallback	元素规定了在 XSL 处理程序不支持 XSL 元素时，所执行的替代代码
xsl:for-each	遍历一个特定的节点集合，循环
xsl:if	仅当 if 条件为真时才会把要应用的模板包含进去
xsl:import	从一个样式表导入内容到另一个样式表，而且一个导入的样式表比原样式表的优先级低
xsl:include	包含一个样式表导入到另一个样式表的内容，而且已包含的样式与原样式具有相同的优先级
xsl:key	声明一个可以在样式表中使用的命名关键字，需要使用 key()函数
xsl:message	输出一个消息
xsl:namespace-alias	把样式表中的命名空间在输出中替换为另一个命名空间

1. <xsl:value-of>元素

<xsl:value-of>元素用于将 XML 文档中指定标记的数据内容进行输出。语法格式如下：

```
<xsl:value-of select="标记匹配模式"/>
```

select 属性指定要提取内容的标记路径。<xsl:value-of>元素可在模板中不限次数的使用。XSLT 处理器首先找到 XSLT 文件的根模板，执行根模板的内容进行 HTML 的转换，当执行到<xsl:value-of>元素时会依据提供的标记匹配模式在 XML 文档中寻找该标记是否存在，如果该标记存在则将数据内容提取出来。

2. <xsl:for-each>元素

<xsl:for-each>元素的作用是循环读取 XML 文档中某一标记的内容。该元素语句格式如下：

```
<xsl:for-each select="标记匹配模式">
  循环内容
</xsl:for-each>
```

XSLT 处理器根据元素中的标记匹配模式到 XML 文档中查找第一个与标记匹配模式匹配的 XML 标记，如果找到则对<xsl:for-each>元素中的内容进行转换，并将转换后的内容放入到 HTML 文件中，然后再查找下一个与标记匹配模式匹配的 XML 标记，直到查找完毕。

3. 控制语句元素

控制语句元素就是在转换过程中能够起到控制语句执行顺序的元素。XSLT 样式表中可以使用两种控制语句元素：<xsl:if>和<xsl:choose>。

介绍两种元素之前，先来介绍经常在控制语句中使用的运算符。在 XSLT 样式表中可以使用两类运算符：比较运算符和逻辑运算符。运算符用于设置相应的筛选条件，进行数据过滤。表 12-12 列出了 XSLT 样式表中可用的比较运算符。

表 12-12　比较运算符

比较运算符	代替符号	说明
=	=	等于
!=	!=	不等于
<	<	小于
<=	<=	小于等于
>	>	大于
>=	>=	大于等于

<xsl:if>元素用于过滤 XML 文档中的内容。当元素中设定的条件为真时，才允许执行其他语句。<xsl:if>元素的语句如下：

```
<xsl:if test="条件表达式">
执行语句或模板
</xsl:if>
```

<xsl:choose>元素、<xsl:when>元素和<xsl:otherwise>元素一起配合使用，用于表示多种条件语句。格式如下：

```
<xsl:choose>
  <xsl:when test="表达式 1">
    执行 1
  </xsl:when>
  <xsl:when test="表达式 2">
    执行 2
  </xsl:when>
……
  <xsl:otherwise>
    执行 n
  </xsl:otherwise>
</xsl:choose>
```

4．<xsl:copy>元素

XSLT 样式表提供<xsl:copy>元素用于将源树结构中的节点复制到结果树结构中，从而不用再为 XML 文档中的每一个元素都提供一个模板。<xsl:copy>元素的语法格式如下：

```
<xsl:copy use-attribute-sets="属性组"></xsl:copy>
```

use-attribute-sets 指定由元素<xsl:attribute-set>创建的属性组。

5．<xsl:sort>元素

<xsl:sort>元素用于将输出结果排序，一般用于<xsl:each>元素中。默认情况下，排序结果是升序。其语法格式如下：

```
<xsl:sort select="标记匹配模式" order=" ascending| descending"></xsl:sort>
```

其中 order 指定排序顺序，ascending 为默认的升序，descending 则为降序。

6．<xsl:element>元素

<xsl:element>元素用于在 XSLT 样式表中创建新的标记，其语法格式为：

```
<xsl:element name="标记名" namespace ="URI" use-attribute-sets ="属性组"/>
```

属性 name 指定要创建的标记名称，属性 namespace 指定标记所属的命名空间，属性 use-attribute-sets 指定使用<xsl:attribute-set>元素创建的属性组。后面两个属性参数在元素声明中是可选的。

7．<xsl:attribute>元素和<xsl:attribute-set>元素

<xsl:attribute>元素用于创建文档中标记的属性，而<xsl:attribute-set>元素则用于创建属性组。xsl:attribute 元素语法格式如下：

```
<xsl:attribute name="" namespace=""/>
```

name 指定创建的属性名称，namespace 指定属性所属的命名空间。<xsl:attribute-set>元素语法格式如下：

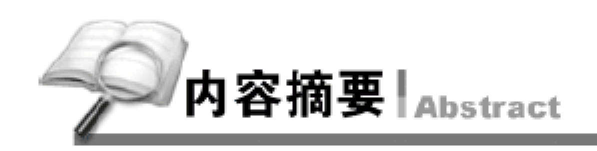
```
<xsl:attribute-set name="" use-attribute-sets="">
  <xsl:attribute name="" namespace=""/>
  ……
</xsl:attribute-set>
```

由于<xsl:attribute-set>元素创建的是一个属性组，所以其中必须使用<xsl:attribute>元素声明组中的属性。



<xsl:attribute-set>元素只能在根元素<xsl:stylesheet>内定义属性组。

第13章 XMLHttpRequest



XMLHttpRequest 对象是 Ajax 核心技术之一，用于实现客户端脚本与服务器之间的数据交互过程。通过使用 XMLHttpRequest 可以从 Web 服务器检索数据，而不必将用户当前正在浏览的页面回馈给服务器，也就是说只更新页面的部分内容而不需要刷新整个页面。

本章首先对 XMLHttpRequest 对象进行简单介绍，然后介绍该对象的属性和方法的使用，接下来介绍如何与服务器进行交互，最后通过示例演示 XMLHttpRequest 对象的使用。

学习目标 Objective

- 理解 XMLHttpRequest 对象
- 掌握 XMLHttpRequest 对象的属性和方法
- 掌握如何发送请求和处理回调函数
- 理解 XMLHttpRequest 对象的运行周期
- 熟练使用 XMLHttpRequest 对象

13.1 XMLHttpRequest 简介

XMLHttpRequest 对象是实现 Ajax 应用必不可少的核心技术，负责 Ajax 与服务器的异步交互。因此 Ajax 应用实现的框架和异步通信的架构类似，也是围绕 XMLHttpRequest 对象的创建、发送请求、处理响应与服务器进行交互。

XMLHttpRequest 对象并非最近才出现，最早在 Microsoft Internet Explorer 5.0 中将 XMLHttpRequest 对象以 ActiveX 组件的方式引入，被称为 XMLHTTP，是一种支持异步请求的技术。

Internet Explorer（Internet Explorer 5.0 以上）允许开发人员在 Web 页面内部使用 XMLHTTP ActiveX 组件扩展自身的功能，并且可以直接传输数据到服务器上或者从服务器获取数据。这个功能很重要，因为它有助于减少无状态连接的等待，还可以排除下载冗余，从而加快进程的速度。

目前，XMLHttpRequest 对象已得到大部分浏览器的支持。Mozilla（Mozilla 1.0 以上及 Netscape 7.0 以上）创建自己的继承 XML 代理类——XMLHttpRequest 类。Konqueror（和 Safari v1.2，同样也是基于 HTML 的浏览器）也支持 XMLHttpRequest 对象，而 Opera 也将在其 7.6x

以后的版本中支持 XMLHttpRequest 对象。对于大多数情况，XMLHttpRequest 对象和 XMLHTTP 组件很相似，方法和属性也类似，只是有一小部分属性不支持。

例如，在 Internet Explorer 中可使用如下代码创建一个 XMLHttpRequest 对象实例。

```
var httpRequest;
httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
```

可以看到，XMLHttpRequest 对象的创建非常容易。具体有关 XMLHttpRequest 对象的属性和方法等内容将在下一节中详细介绍。

13.2　XMLHttpRequest 成员

XMLHttpRequest 对象创建好之后，就可以调用该对象的方法和属性进行异步传输数据。下面介绍 XMLHttpRequest 对象的各种属性和方法，以便于脚本处理和控制 HTTP 请求与响应。

13.2.1　XMLHttpRequest 属性

首先介绍该对象的属性，通过属性可以判断请求的状态、返回信息的格式等。如表 13-1 所示，下面将详细介绍这些属性。

表 13-1　XMLHttpRequest 对象属性

属性	说明
onreadystatechange	指定当 readyState 属性改变时的事件处理句柄，只写
readyState	返回当前请求的状态，只读
responseBody	将响应信息正文以 unsigned byte 数组的形式返回，只读
responseStream	以 ADO Stream 对象的形式返回响应信息，只读
responseText	将响应信息作为字符串返回，只读
responseXML	将响应信息格式化为 Xml Document 对象并返回，只读
status	返回当前请求的 HTTP 状态码，只读
statusText	返回当前请求的响应行状态，只读

1．onreadystatechange 属性

onreadystatechange 属性表示当每个状态改变时都会触发这个事件处理器，通常会调用一个 JavaScript 函数，即当服务器返回数据到客户端时需要相应的 JavaScript 函数处理这些数据，而这个函数需要使用该属性定义。

2．readyState 属性

readyState 用于表示请求的状态。有 5 个可取值，如下所示。

- **0**　表示未初始化，即对象已建立，但是尚未初始化（尚未调用 open()方法）。

- 1 表示正在加载，此时对象已建立，尚未调用 send()方法。
- 2 表示已加载，即 send()方法已调用，但是当前的状态及 HTTP 头未知。
- 3 表示交互中，此时已接收部分数据，因为响应及 HTTP 头不全，这时通过 responseBody 和 responseText 获取部分数据时会出现错误。
- 4 表示完成，即数据接收完毕，此时可以通过 responseBody 和 responseText 获取完整的回应数据。

3．responseBody 属性

当服务器返回请求的状态为完成时，可以通过此属性来获得接收的数据，该属性得到的是二进制的 byte 流。语法如下所示：

```
byte[] strValue=oXMLHttpRequest.responseBody
```

4．responseStream 属性

以 ADO Stream 对象的形式返回响应信息，语法如下所示：

```
object strValue=XMLHttpRequest.responseStream
```

5．responseText 属性

表示服务器的文本响应，其处理结果以字符串形式返回，XMLHttpRequest 默认将响应数据的编码格式定为 UTF-8。如果服务器返回的是 XML 文档，此属性并不处理 XML 文档中的编码声明，需要使用 responseXML 来处理。

6．responseXML 属性

表示服务器的响应，其结果以 XML 文件格式返回。这个对象可以解析为一个 DOM 对象，即将响应信息格式化为 XmlDocument 对象并返回。如果响应数据不是有效的 XML 文档，属性本身不返回 XMLDOMParseError，可以通过处理过的 DOMDocument 对象获取错误信息。

7．status 属性

指明请求是否已成功，还可以显示请求失败的具体原因。返回值为一个 3 位数，格式是 1XX～5XX，但不是中间的所有数字。表 13-2 中对常用的属性值做了简要说明。

表 13-2 status 属性值的含义

status 属性值	说明	例子
100～101	这类状态代码表示临时的响应。客户端在收到常规响应之前，应准备接收一个或多个 1XX 响应	100 表示继续：初始的请求已经接受，客户应当继续发送请求的其余部分
200～206	这类状态代码表明服务器成功的接受客户端请求	200 表示一切正常，对 GET 和 POST 请求的应答文档跟在后面
300～307	客户端浏览器必须采取更多操作来实现请求。例如，浏览器可能不得不请求服务器上不同的页面，或通过代理服务器重复该请求	300 表示多种选择：客户请求的文档可以在多个位置找到，这些位置已经在返回的文档内列出。如果服务器要提出优先选择，则应该在 Location 应答头指明

续表

status 属性值	说明	例子
400～423	发生错误，客户端有问题。例如，客户端请求不存在的页面，客户端未提供有效的身份验证信息	404 资源未找到：无法找到指定位置的资源。这也是一个常用的应答
500～505	服务器由于遇到错误而不能完成该请求	505 HTTP 版本不支持：服务器不支持请求中所指明的 HTTP 版本

statusText 属性描述 HTTP 状态代码文本，并且仅当 readyState 值为 3 或 4 时才可用。当 readyState 为其他值时，试图存取 statusText 属性将引发一个异常。

13.2.2　XMLHttpRequest 方法

可以通过属性的值判断本次 XMLHttpRequest 的进展情况和结果。如果需要操作 XMLHttpRequest 对象，还需要了解 XMLHttpRequest 对象的方法，如表 13-3 所示。

表 13-3　XMLHttpRequest 对象方法

方法	说明
open()	创建一个新的 HTTP 请求，并指定请求的方法、URL 以及验证信息
send()	发送请求到 HTTP 服务器并接收回应
setRequestHeader()	单独指定请求的某个 HTTP 头
getAllResponseHeaders()	获取响应的所有 HTTP 头
getResponseHeader()	从响应信息中获取指定的 HTTP 头
abort()	取消当前请求

1. open()方法

open()方法的完整语法格式如下：

```
open(string method, string url, boolean async, string username, string password)
```

其中，method 是必选参数，用于指定发送请求的 HTTP 方法（GET、POST、PUT、DELETE 或 HEAD），为了将数据发送到服务器应该使用 POST 方法；但若要从服务器端检索数据应该使用 GET 方法。另外，url 参数用于指定 XMLHttpRequest 对象将请求发送到的服务器相应的 url。async 参数指定是否请求是异步的，默认值为 true，在发送同步请求时需要将这个参数设置为 false。对于需要认证的服务器，可以提供可选的用户名和口令参数。

在调用 open()方法准备好一个请求之后，还需要将该请求发送到服务器。仅当 readyState 的值为 1 时才可以调用 send()方法，否则 XMLHttpRequest 对象将引发一个异常。该请求使用提供给 open()方法的参数发送到服务器。当 async 参数值为 true 时，send()方法立即返回，从而允许其他客户端脚本处理继续。在调用 send()方法后，XMLHttpRequest 对象将 readyState 的值设置为 2。当服务器响应时，在接收消息之前如果存在任何消息，XMLHttpRequest 对象将把 readyState 的值设置为 3。当请求完成加载时，把 readyState 的值设置为 4。对于一个 HEAD 类型的请求，将在把 readyState 的值设置为 3 后再立即设置为 4。

2．send()方法

send()方法包含一个可选的参数，该参数可以包含可变类型的数据。典型的应用是通过 POST 方法将数据发送到服务器。另外也可以显式地调用无参数 send()方法，对于大多数情况，在调用 send()方法之前应该使用 setRequestHeader()方法设置 Content-Type 头部。如果在 send(data)方法中的 data 参数的类型为 string，那么数据将被编码为 UTF-8。如果数据是 Document 类型，那么将使用由 data.xmlEncoding 指定的编码串行化该数据。

例如，向 myphp.php 文件以异步方式发送一个 GET 请求，便可以使用如下的代码：

```
xmlHttp.open("GET","myphp.php",true);
xmlHttp.send(null);
```

现在同样需要向 myphp.php 文件发送请求，不同的是要求请求中带有一个字符串 name=Anne。实现这个有两种方法，第一种是使用 open()方法在 url 参数中指定：

```
xmlHttp.open("GET"," myphp.php?name=Anne",true);
xmlHttp.send(null);
```

第二种是使用 send()方法指定：

```
xmlHttp.open("POST"," myphp.php ",true);
xmlHttp.send("name=Anne");
```

3．setRequestHeader()方法

```
void setRequestHeader(string header, string value)
```

setRequestHeader()方法为 HTTP 请求中一个给定的首部设置值。该方法有两个参数，第一个参数 header 字符串表示要设置的首部，第二个参数 value 字符串表示要在首部中放置的值，这个方法必须在 open()之后才能调用。

4．getAllResponseHeaders()方法

```
string getAllResponseHeaders()
```

Web 应用开发人员对该方法的核心功能应该很熟悉。它用于返回一个字符串，其中包含 HTTP 请求的所有响应首部。首部包括 Content-Length、Date 和 URL。

5．getResponseHeader()方法

该方法与 getAllResponseHeaders()是对应的，不过它用一个参数来表示希望得到哪一个首部值，并且会把这个值作为一个字符串返回。

```
string getResponseHeader(string header)
```

6．abort()方法

调用 abort()方法取消当前请求，调用该方法后当前请求返回 UnInitialized 状态。语法如下

所示：

```
XMLHttpRequest.abort()
```

在这些方法中，最常用的方法就是 open()和 send()。XMLHttpRequest 对象还有许多属性，在设计 Ajax 交互时这些属性非常有用。

13.3　XMLHttpRequest 与服务器通信

XMLHttpRequest 与服务器通信有两种方式：同步方式和异步方式。同步方式的调用非常简单，仅仅适用于数据量非常少的场合。如果数据量很大，会造成用户界面很长一段时间的停顿，这当然会损害 Web 应用的可用性。为了不影响可用性，Ajax 应用中一般都使用异步方式来与服务器通信。用于设置同步和异步方式的是 XMLHttpRequest 对象 open()方法的 async 参数，其值为 true 代表异步，为 false 代表同步。

13.3.1　创建 XMLHttpRequest 对象

在使用 XMLHttpRequest 对象发送请求和处理响应之前，首先需要使用 JavaScript 创建一个 XMLHttpRequest 对象。由于 XMLHttpRequest 不是一个 W3C 标准，所以可以采用多种方法创建 XMLHttpRequest 的实例。Internet Explorer 将 XMLHttpRequest 实现为一个 ActiveX 对象，其他浏览器（如 Firefox、Safari 和 Opera）实现为一个本地 JavaScript 对象。由于存在这些差别，JavaScript 代码中必须包含相关逻辑，从而使用 ActiveX 对象或者使用本地 JavaScript 对象技术来创建 XMLHttpRequest 的一个实例。

但是，并不需要针对每个浏览器详细编写代码来区别浏览器类型。要创建一个 XMLHttpRequest 对象的实例，需要做的只是检查浏览器是否提供对 ActiveX 对象的支持。如果浏览器支持 ActiveX 对象，就可以使用 ActiveX 来创建 XMLHttpRequest 对象。否则就要使用本地 JavaScript 对象技术来创建。下面代码中的 createXMLHttpRequest()函数演示使用 JavaScript 代码来编写跨浏览器的 XMLHttpRequest 对象实例，如代码 13.1 所示。

代码 13.1　创建函数：createXMLHttpRequest()

```
<script type="text/javascript">
var xmlHttp;
function createXMLHttpRequest()
{
  //在 IE 下创建 XMLHttpRequest 对象
  try
  {
    xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
  }
  catch(e)
```

```
    {
      try
      {
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
      }
    catch(oc)
    {
      xmlHttp = null;
    }
   }
    //在 Mozilla 和 Safari 等非 IE 浏览器下创建 XMLHttpRequest 对象
    if(!xmlHttp && typeof XMLHttpRequest != "undefined")
   {
    xmlHttp = new XMLHttpRequest();
   }
  return xmlHttp;
  }
  </script>
```

在上述代码中，首先创建一个全局变量 xmlHttp 来保存这个对象的引用。createXMLHttpRequest()函数完成创建 XMLHttpRequest 实例的具体工作。这个方法中使用简单的分支逻辑（选择逻辑）确定如何创建对象。对 window.ActiveXObject 的调用会返回一个对象，也可能返回 null。if 语句将调用返回的结果看作是 true 或 false（如果返回对象则为 true，返回 null 则为 false），以此指示浏览器是否支持 ActiveX 对象，相应地得知浏览器是不是 Internet Explorer。如果是则通过实例化 ActiveXObject 的一个新实例来创建 XMLHttpRequest 对象，并传入一个字符串指示要创建何种类型的 ActiveX 对象。如果调用失败，则会尝试使用 Microsoft.XMLHTTP 创建。最后，如果两种情况均不成功则说明使用的是非 Internet Explorer 浏览器，此时会将 XMLHttpRequest 实现为一个本地 JavaScript 对象并返回。

由于 JavaScript 具有动态类型特性，而且 XMLHttpRequest 在不同浏览器上的实现兼容，所以可以使用同样的方式访问 XMLHttpRequest 实例的属性和方法，而不需要考虑实例创建的方法是什么。这就简化了开发过程，而且在 JavaScript 中也不必编写特定于浏览器的逻辑处理。

13.3.2 发送请求

XMLHttpRequest 对象可以使用 GET 或 POST 方法发送请求，请求数据可以作为字符串、XML 或者 JSON 数据发送。处理请求之后，服务器一般会发送简单文本、XML 数据或 JSON 数据作为响应。

下面使用代码 13.1 中创建的 createXMLHttpRequest()函数创建一个 xmlHttp 对象实例，向服务器发送请求。要实现这一操作，需要使用 XMLHttpRequest 对象的 open()方法和 send()方法。假设有一个名为 simpleResponse.xml 文件具有如代码 13.2 所示内容。

代码 13.2　simpleResponse.xml 文件内容

```
<?xml version="1.0" encoding="GB2312"?>
<Message>
this is XMLHttpRequest
</Message>
```

使用 GET 和 POST 方法如下：

```
//使用 GET 方法
xmlHttp.open("GET", "simpleResponse.xml",true);
xmlHttp.send(null);
//使用 POST 方法
xmlHttp.open("POST","simpleResponse.xml",true);
xmlHttp.send(user.xml);
```

例如在一个 HTML 页面中只有一个按钮，页面源程序如代码 13.3 所示。单击这个按钮会初始化一个发至服务器的异步请求。服务器将返回一个简单的静态文本文件作为响应。在处理这个响应时，会在一个警告窗口中显示静态文本文件的内容。

代码 13.3　提交给服务器端的 HTML 代码

```
<form action="#">
<input type="button" value="check me" onclick="startRequest();"/>
</form>
```

当单击按钮触发 onclick 事件时调用 startRequest()函数的代码。该函数用于向服务器发送请求，如代码 13.4 所示。

代码 13.4　startRequest()函数

```
//发送请求
function startRequest() {
createXMLHttpRequest();  //创建 XMLHttpRequest 对象的实例
xmlHttp.onreadystatechange = handleStateChange;
xmlHttp.open("GET", "simpleResponse.xml", true);
xmlHttp.send(null);
}
```

在上述代码中使用 createXMLHttpRequest()函数创建 XMLHttpRequest 对象的实例，然后使用 onreadystatechange 属性指定回调函数，然后使用 open()和 send()方法完成发送请求操作。

13.3.3　处理回调函数

当请求状态改变时 XMLHttpRequest 对象调用 onreadystatechange 注册事件处理器。因此在处理该响应之前，事件处理器应该首先检查 readyState 的值和 HTTP 状态。当请求完成加载（readyState 的值为 4）并且响应已经完成（HTTP 状态为 OK）时，就可以调用一个 JavaScript

函数来处理该响应内容。

在代码 13.4 中创建的 startRequest()函数中的 onreadystatechange 属性指定 handleStateChange()函数，该函数用于接收服务器返回的数据，并根据需求动态地更新页面（一般使用 DOM 完成）或者进行一些其他的操作（例如，判断并弹出提示对话框）。handleStateChange()函数如代码 13.5 所示。

代码 13.5　handleStateChange()函数

```
function handleStateChange()
{
if(xmlHttp.readyState == 4)
 {
if(xmlHttp.status == 200)
  {
alert("The server replied with: " + xmlHttp.responseText);
}
else{
alert("错误，请求页面异常!");
}
 }
}
```

将上述代码保存为 xmlhttp.html，然后在浏览器地址栏中输入“http://localhost:8080/xmlhttp.html”。在页面中可以看到一个按钮，当单击 check me 按钮时则会弹出一个对话框，并在对话框中将 XML 文档中的信息显示出来，运行结果如图 13-1 所示。

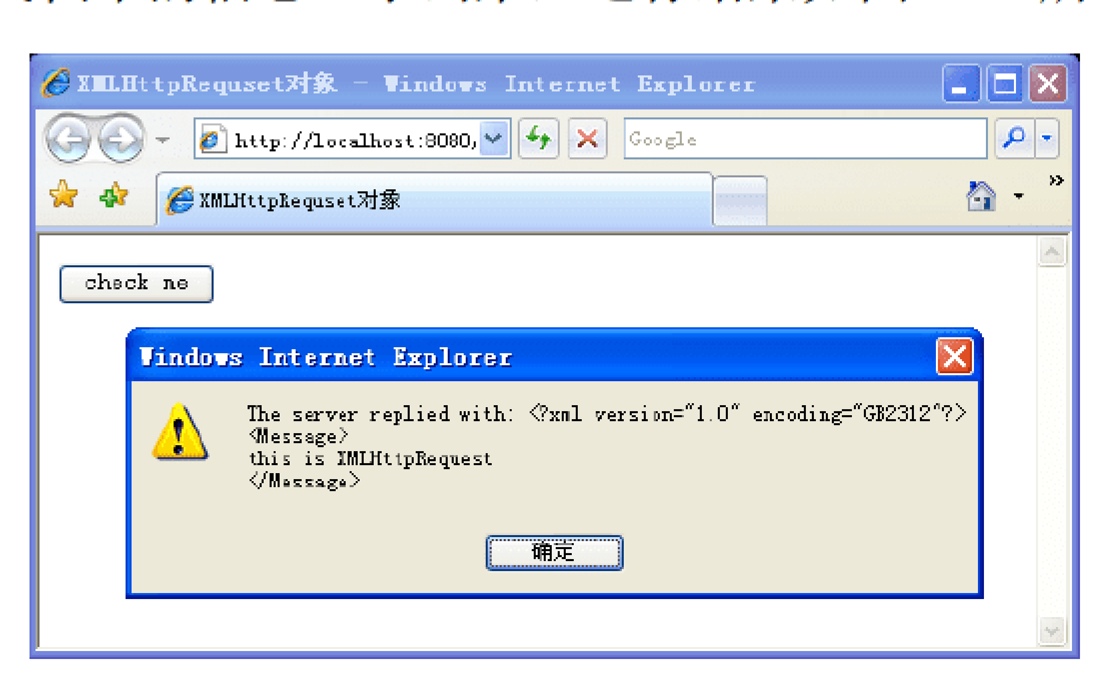


图 13-1　运行结果

13.4　XMLHttpRequest 对象运行周期

Ajax 程序主要通过 JavaScript 事件来触发，在运行时需要调用 XMLHttpRequest 对象发送请求和处理响应，客户端处理完响应之后 XMLHttpRequest 对象就会一直处于等待状态，这样一直周而复始的工作。

Ajax 实质上是遵循“客户端/服务器端”模式，所以这个框架基本流程是：XMLHttpRequest 对象初始化→发送请求→服务器接收→服务器返回→客户端接收→修改客户端页面内容。只不过这个过程是异步的，其周期如图 13-2 所示。

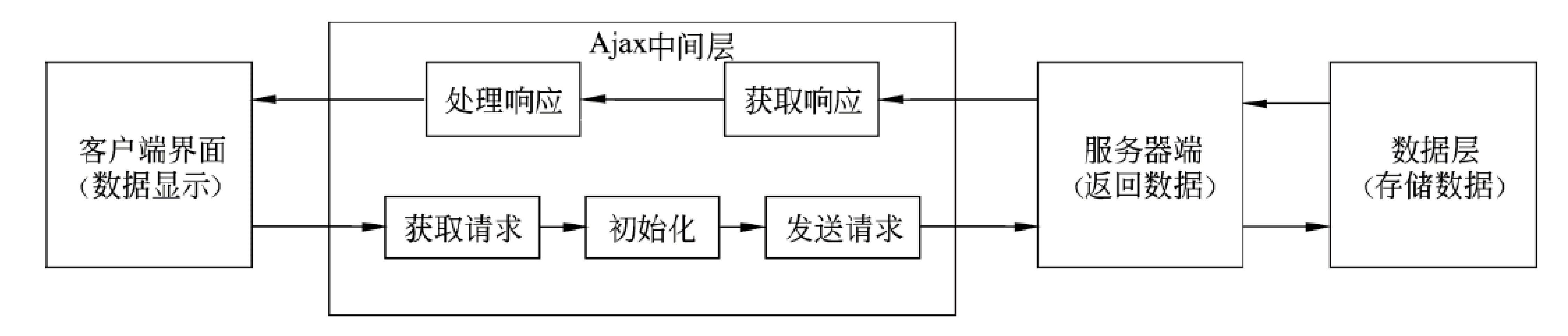


图 13-2　XMLHttpRequest 对象运行周期

在图 13-2 中，Ajax 中间层显示 XMLHttpRequest 对象的运行周期。当 Ajax 中间层从客户端界面获取请求信息之后需要初始化 XMLHttpRequest 对象，初始化完成之后通过 XMLHttpRequest 对象将请求发送给服务器端。服务器端获取请求信息后处理并返回响应信息。然后 Ajax 中间层获取响应，通过 XMLHttpRequest 对象将响应信息和 Ajax 中间层所设置的样式信息进行组合，即处理响应。最后 Ajax 中间层将所有的信息发送给客户端界面，并显示由服务器返回的信息。

1．初始化 XMLHttpRequest 对象

为了让 JavaScript 可以向服务器发送 HTTP 请求，必须使用 XMLHttpRequest 对象。使用之前需要将 XMLHttpRequest 对象实例化。之前说过，各个浏览器对这个实例化过程实现不同。Internet Explorer 以 ActiveX 组件的形式提供，而 Mozilla 等浏览器则直接以 XMLHttpRequest 的形式提供。为了让编写的程序能够跨浏览器运行，可以使用代码 13.1 创建的跨浏览器的 XMLHttpRequest 对象实例。

2．指定响应处理函数

接下来需要指定当服务器返回信息时客户端的处理方式。只要将相应的处理函数名称赋给 XMLHttpRequest 对象的 onreadystatechange 属性就可以。例如：

```
xmlHttp.onreadystatechange = processRequest;
```

需要指出的是，这个函数名称不加括号、不指定参数。也可以使用 JavaScript 即时定义函数的方式定义响应函数。例如：

```
xmlHttp.onreadystatechange = function() {};
```

3．发出 HTTP 请求

指定响应处理函数之后就可以向服务器发出 HTTP 请求，这一步调用 XMLHttpRequest 对象的 open()和 send()方法。

```
xmlHttp.open('GET', 'http://www.example.org/some.file', true);
xmlHttp.send(null);
```

按照顺序，open()方法调用完毕之后需要调用 send()方法。send()方法的参数如果是以 POST 方式发出的话，可以是任何想传给服务器的内容。不过跟 form 表单一样，如果要传送文件或者 POST 内容给服务器，必须先调用 setRequestHeader()方法修改 MIME 类别。如下所示：

```
xmlHttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
```

这时发送信息则以查询字符串的形式列出，作为 send()方法的参数发送。其代码如下所示：

```
name=value&anothername=othervalue&so=on
```

4．处理服务器返回的信息

在第二步已经指定响应处理函数，接下来讲解这个响应处理函数都应该做什么。首先需要检查 XMLHttpRequest 对象的 readyState 属性的值，判断请求目前的状态。readyState 的值为 4 的时候代表服务器已经传回所有的信息，可以开始处理信息并更新页面内容。如下：

```
if (xmlHttp.readyState == 4) {
// 信息已经返回，开始处理
} else {
// 信息还没有返回，等待
}
```

服务器返回信息后还需要判断返回的 HTTP 状态码，确定返回的页面没有错误。所有的状态码都可以在 W3C 的官方网站上查到，其中 200 代表页面正常。

```
if xmlHttp.status == 200) {
// 页面正常，可以开始处理信息
} else {
// 页面有问题
}
```

XMLHttpRequest 对象成功返回的信息通常用以下两种方式处理。

- **responseText**　将返回的信息当字符串使用。
- **responseXML**　将返回的信息当 XML 文档使用，可以用 DOM 文档对象模型处理。

上述 4 个步骤就是 XMLHttpRequest 对象在 Ajax 程序中的运行周期，即初始化->指定响应函数->发送请求->处理响应，实际上也是 Ajax 程序进行异步传输数据的基础。

13.5　XMLHttpRequest 实例

前面已经介绍如何创建 XMLHttpRequest 对象，以及发送请求和处理请求的方法。本节将通过 3 个示例演示 XMLHttpRequest 对象在 Ajax 技术中的具体应用。

13.5.1　局部刷新

在本示例中演示如何通过 Ajax 技术实现局部刷新页面，改善用户体验。该例子演示当用户选择列表框中的信息之后，单击【立即查询】按钮时在不刷新页面的情况下显示选择的信息和当前系统时间。下面是实现操作的具体步骤。

首先在 HTML 页面中添加一个 form 表单，在表单中添加一个选择控件、一个超链接按钮和一个 table 表格，如代码 13.6 所示。

代码 13.6　HTML 代码

```
<form action="#">
 <select name="number" id="number">
  <option value="2222222">2222222</option>
  <option selected="selected" value="7777777">7777777</option>
  </select>
  <a href="#" onclick="showRoles('test','number')">立即查询</a>
  <table>
  <tr style="display:none ">
  <td id="test" height="20"> </td>
  </tr>
  </table>
</form>
```

在上述代码中添加一个选择控件，默认选择第二个选项。当用户单击【立即查询】按钮触发 onclick 事件后调用 showRoles()函数，在该函数中传递两个形参，并且在 id 为 test 的表格中显示由服务器返回的信息。

接下来需要创建 XMLHttpRequest 对象，创建该对象如代码 13.1 所示，在这里不再多述。对象创建好以后需要向服务器发送请求，发送请求的函数如代码 13.7 所示。

代码 13.7　send_request()函数

```
//发送请求
function send_request(url)
{
  createXMLHttpRequest();
  xmlHttp.onreadystatechange = processRequest;
  xmlHttp.open("GET", url, true);
  xmlHttp.send(null);
}
```

在上述代码中为函数传递 url 参数，接下来创建 XMLHttpRequest 对象。语句“xmlHttp.open("GET", url, true)”用于建立对服务器的调用。处理完成之后使用 processRequest()函数处理返回信息。

在客户端使用 xmlHttp 发送请求后，每次状态改变都会调用 onreadystatechange 属性的

processRequest()函数，该函数的具体实现如代码 13.8 所示。

代码 13.8　processRequest()函数

```
//处理返回信息的函数
function processRequest()
  {
      if (xmlHttp.readyState == 4)
      {
      //判断对象状态
          if (xmlHttp.status == 200)
          {
          //信息已经成功返回，开始处理信息
          document.getElementById(currentPos).innerHTML = xmlHttp.response
          Text;}
          else
          {   //页面不正常
          alert("您所请求的页面有异常。");
          }
      }
  }
```

回调函数创建好以后需要再创建一个函数，用于处理请求。如果成功建立请求之后，则会显示一个提示信息“正在读取数据…”，处理请求的函数如代码 13.9 所示。

代码 13.9　showRoles()函数

```
function showRoles(obj,obj2)
  {
  currentPos = obj;
  document.getElementById(obj).parentNode.style.display = "";
  document.getElementById(obj).innerHTML = "正在读取数据..."
  send request("Server.aspx?phonenum="+document.getElementById(obj2).
  value+"&r"+Math.random());
  }
```

在上述代码中，showRoles()函数首先使用 DOM 获取选择的值，然后为 send_request()函数传递参数。Math.random()函数的作用是用于产生一个随机数，解决由缓存问题引起的数据守旧。最后需要创建服务器页面 telephone.php 页面，该页面如代码 13.10 所示。

代码 13.10　telephone.php

```
<?php
$phonenum = $_GET['phonenum'];
$date =getdate();
echo $phonenum;
echo "|";
```

```
    echo $date['year']."-".$date['month']."-".$date['day'];
    echo " ";
    echo $date['hours'].":".$date['minutes'];
    ?>
```


在 PHP 页面中，首先定义一个变量用于获取客户端传递的参数，接着定义一个变量用于获取当前系统的时间，最后输出相关信息。

至此，程序已经全部设计完成。保存代码，运行程序查看结果。在浏览器地址栏中输入“http://localhost:8080/telephone.html”，单击【立即查询】按钮之后成功建立客户端与服务器端的请求，显示提示信息“正在读取数据…”，如图 13-3 所示。当请求完成之后显示服务器返回的信息，如图 13-4 所示。当用户再次单击【立即查询】按钮之后，可以在页面不刷新的情况下更改显示时间。

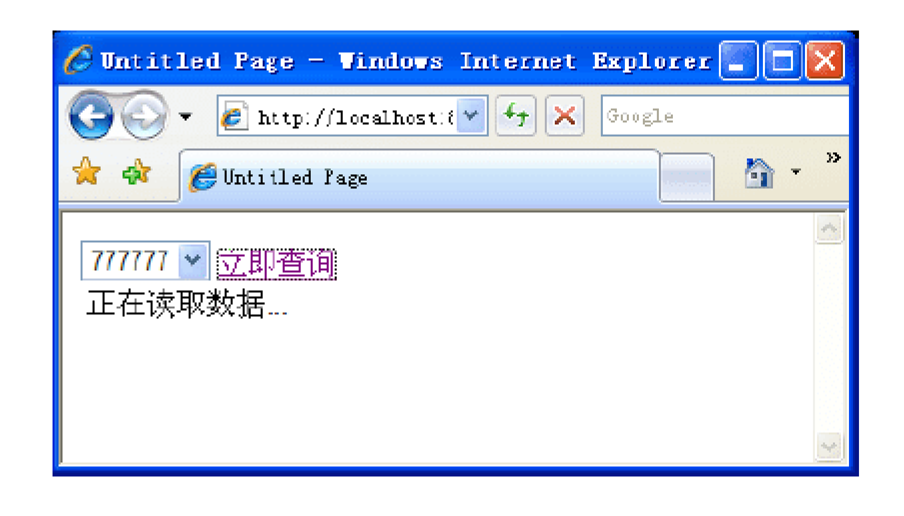


图 13-3　发送请求时

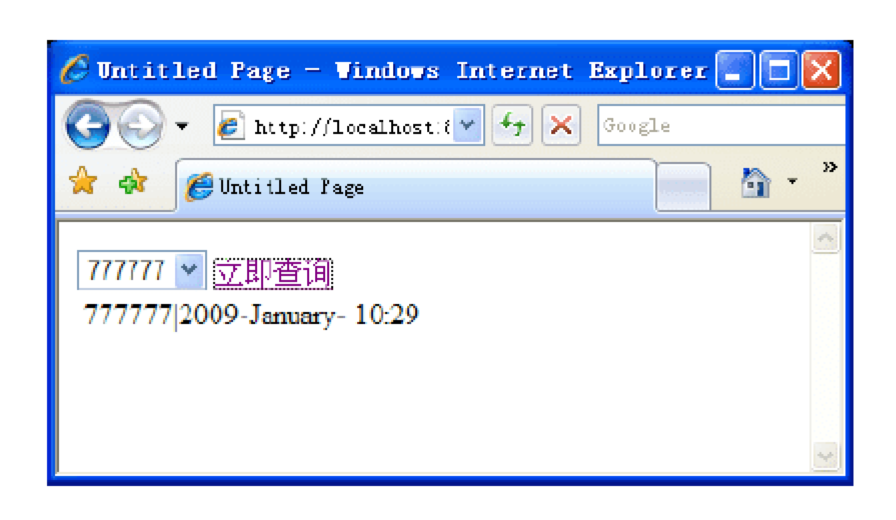


图 13-4　处理结果

13.5.2　操作 XML

在上面的例子中主要讲解如何使用 Ajax 提交请求、传递参数、服务器端处理请求并返回值，以及在客户端的回调函数中显示结果。使用这种方式会每次向服务器发送数据并处理响应，在请求频繁的页面中使用则会出现问题，这与服务器请求的次数相关。因此，如何减少对服务器的请求次数来节省性能成为需要解决的问题。

下面通过将客户端模型作为 XML 发送到服务器，由于 XML 本身的优势及浏览器支持的特性可以将客户端模型作为 XML 格式，然后将其作为 HTTP 请求的一部分发送到服务器，服务器再对 XML 进行处理返回相应的结果。首先需要创建一个 XML 文档，如代码 13.11 所示。

代码 13.11　data.xml

```
<?xml version="1.0" encoding="GB2312" ?>
<mailbox>
  <mail>
    <to>简凝</to>
    <from>江南</from>
    <heading>好久不见！</heading>
    <body>好久不见，最近还好吗？</body>
  </mail>
  <mail>
```

```
        <to>江南</to>
        <from>简凝</from>
        <heading>周末有空么? </heading>
        <body>周末有空么? 一起吃饭吧! </body>
    </mail>
</mailbox>
```

接下来创建一个 xmlExplem.html 作为客户端的页面。在该页面中添加一个按钮和一个<div>标记，<div>标记用于显示 XML 文件的信息。部分代码如代码 13.12 所示。

代码 13.12　xmlExplem.html 部分代码

```
<form action="#">
  <p><button onclick="startRequest()">查看信息</button></p>
  <div id="message"></div>
</form>
```

页面布局好以后，接下来需要创建 XMLHttpRequest 对象。创建 XMLHttpRequest 对象的函数如代码 13.1 所示。然后需要创建一个 startRequest()函数，该函数主要用于创建 XMLHttpRequest 对象、发送请求以及指定回调函数，如代码 13.13 所示。

代码 13.13　startRequest()函数

```
//发送请求
function startRequest()
{
    createXMLHttpRequest();
    xmlHttp.onreadystatechange = parseMailBox;
    xmlHttp.open("GET", "data.xml", true);
    xmlHttp.send(null);
}
```

最后创建一个回调函数 parseMailBox()。该函数的主要功能是解析 XML 文档，并动态改变内容按格式添加到页面上，如代码 13.14 所示。

代码 13.14　parseMailBox()函数

```
function parseMailBox()
  {
      if (xmlHttp.readyState == 4)
      {
      //判断对象状态
          if (xmlHttp.status == 200)
          {
            var mailBoxXML = xmlHttp.responseXML;
            //获取所有信件
            var mails = mailBoxXML.getElementsByTagName('mail');
```

```
                var mailStr = "";
            }
            for (var i = 0; i < mails.length; i++)
            {
                //读取 XML 节点中的数据
                var to = mails[i].getElementsByTagName("to")[0].firstChild.data;
                var from =mails[i].getElementsByTagName("from")[0].firstChild.
                data;
                var heading =mails[i].getElementsByTagName("heading")[0].first
                Child.data;
                var body =mails[i].getElementsByTagName("body")[0].
                firstChild.data;
                //为信件添加附属信息
                mailStr += "收信人: " + to + "<br>";
                mailStr += "发信人: " + from + "<br>";
                mailStr += "标题: " + heading + "<br>";
                mailStr += "正文: <br>" + body + "<br><br>";
            }
            document.getElementById("message").innerHTML = mailStr;
        }
    }
```

至此，程序已经设计完成。将 XML 文件和 HTML 文件保存在同一目录下。运行程序，查看运行结果。页面运行成功后单击【查看信息】按钮，则会将 XML 中的信息读取到页面中。运行结果如图 13-5 所示。

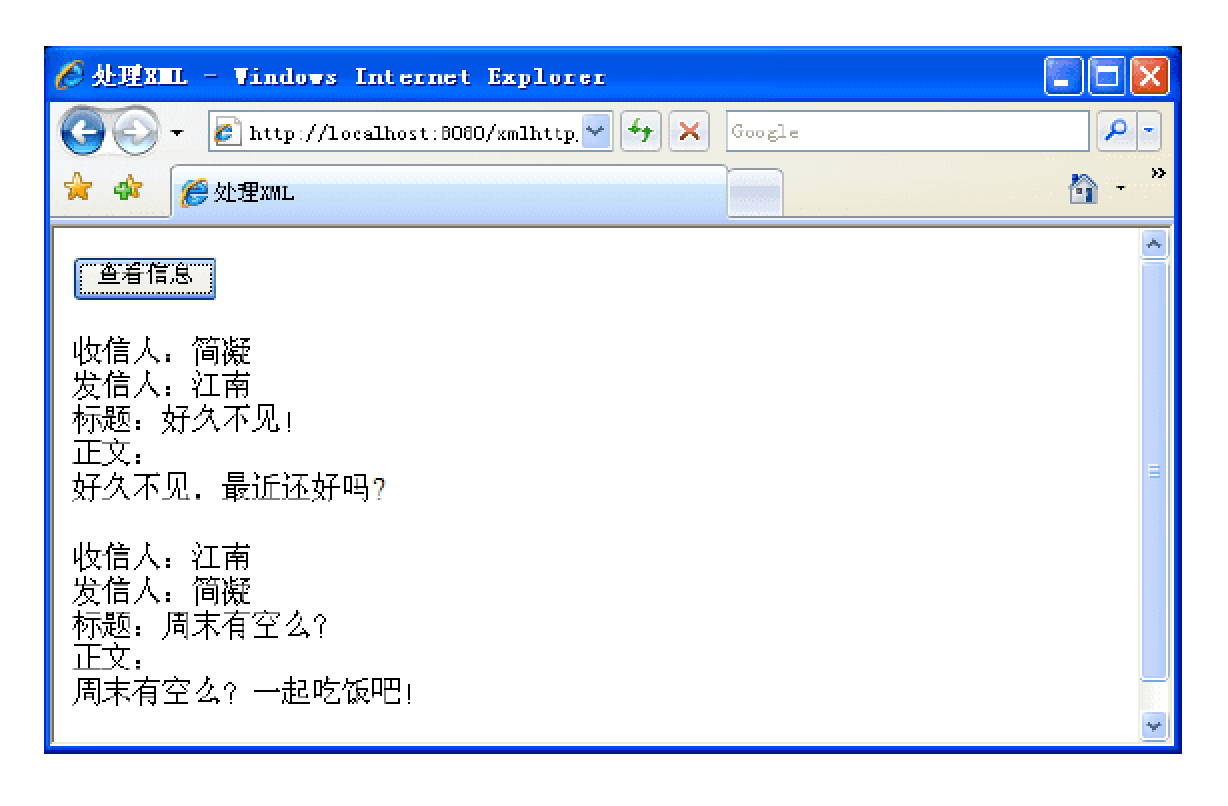


图 13-5 处理 XML

13.5.3 级联菜单

遇到需要动态加载或者更新下拉列表框的问题，如果使用传统的 Web 技术就需要频繁的通过刷新页面的方法来解决。而按需读取数据是 Ajax 技术最大的特色，这样可以有效减少数据冗余，并且减轻服务器的负担，同时还可以提高带宽的利用率，非常适合制作 Web 中的分

类树或树形结构。

下面通过应用 Ajax 技术来更好地实现动态加载或更新下拉列表框的功能。首先创建静态页面，该页面提供动态加载下拉列表框的界面，具体实现如代码 13.15 所示。

代码 13.15　页面局部代码

```
<form id="form1" runat="server">
    <div>
      <h3> 动态加载列表框</h3>
    </div>
    省份:
    <select id="province" onchange="SendRequest(this.value)">
        <option value="0">-选择省份-</option>
        <option value="bj">北京</option>
        <option value="sh">上海</option>
        <option value="hb">湖北</option>
        <option value="js">江苏</option>
    </select>
      城市:
    <select id="city">
        <option value="0">-选择城市-</option>
    </select>
</form>
```

在上述代码中，主要创建两个下拉列表框 province 和 city。province 在 onchange 事件触发时执行 SendRequest()函数，将用户选择的值作为参数发送到服务器。

然后创建 SendRequest()函数，该函数主要用于创建 XMLHttpRequest 对象、发送请求和指定回调函数，如代码 13.16 所示。

代码 13.16　SendRequest()函数

```
//发送请求
function SendRequest(str)
{
    var url = "jiliancaidan.php?sort="+escape(str);
    createXMLHttpRequest();
    XmlHttp.onreadystatechange = handleStateChange;
    XmlHttp.open( "GET", url, true );
    XmlHttp.send( null );
}
```

SendRequest()函数将在下拉列表框中选择的值 str 作为 sort 参数提交到服务器页面，再指定处理由服务器返回信息的回调函数。

接下来创建回调函数，在 HTML 中定义回调函数处理服务器返回的字符串。即将字符串按逗号分割为多个字符串，再将这些字符串动态生成<option>标记添加到城市下拉列表框中，

如代码 13.17 所示。

代码 13.17　handleStateChange()函数

```
//回调函数
function handleStateChange()
{
   if( XmlHttp.readyState == 4 )
    {
     if(XmlHttp.status == 200 )
        {
               var citys=XmlHttp.responseText;
               var citya=citys.split(",");
               while(document.getElementById("city").options.length>0)
              {
                 //将列表框清空
                 document.getElementById("city").options.remove(0);
              }
              for(var j=0;j<citya.length;j++)
                 {
                 var oOption = document.createElement("option");//生成 option 对象
                 oOption.text=citya[j];
                 oOption.value=citya[j];
                 document.getElementById("city").add(oOption);//把 option 对象加入
                }
      }
         else
                {
           alert("请求页面异常!");
         }
    }
}
```

最后再创建一个 jiliancaidan.php 服务器页面，在该页面中处理客户端发送的请求。jiliancaidan.php 页面如代码 13.18 所示。

代码 13.18　jiliancaidan.php

```
<?php
 header('Content-type:text/html;charset=GBK');
 $sort =$_GET['sort'];
 $result ="";
 if($sort !=="")
 {
   switch($sort)
   {
```

```
        case "bj":
        $result="海淀区,天安门";
        break;
        case "sh":
        $result="浦东,浦西";
        break;
        case "hb":
        $result="武汉,宜昌";
        break;
        case "js":
        $result="南京,扬州";
        break;
      }
    echo $result;
  }
?>
```

此时程序已经设计完成，接下来在浏览器中查看运行效果。初始化时会在【省份】下拉列表框中显示“-选择省份-”项，而【城市】下拉列表框中显示“-选择城市-”，如图 13-6 所示。当在选择省份之后，城市列表框中的内容将相应改变，图 13-7 所示为选择“北京”后动态加载的列表。

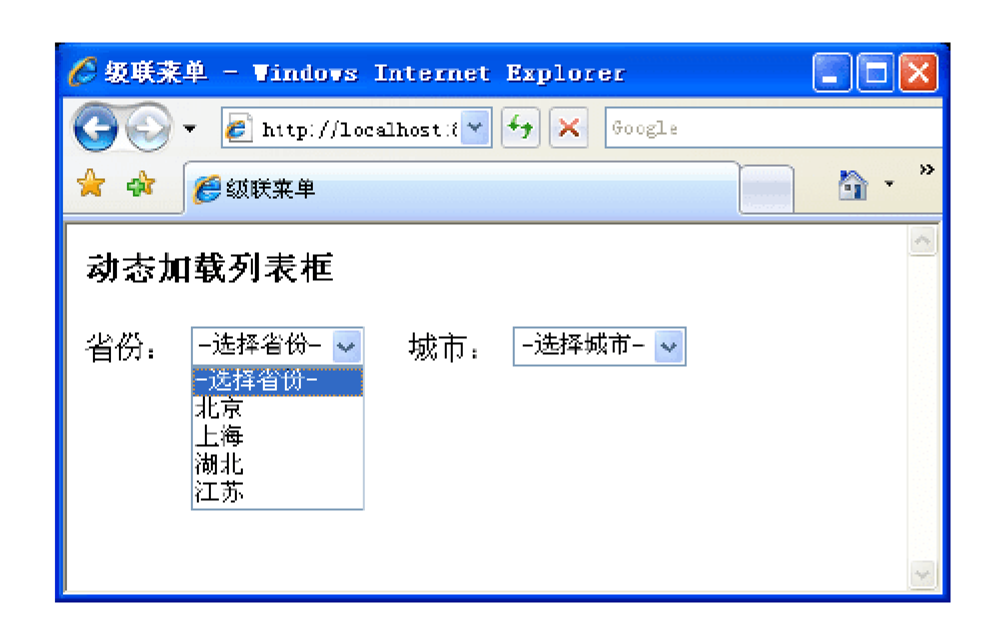


图 13-6　初始化结果

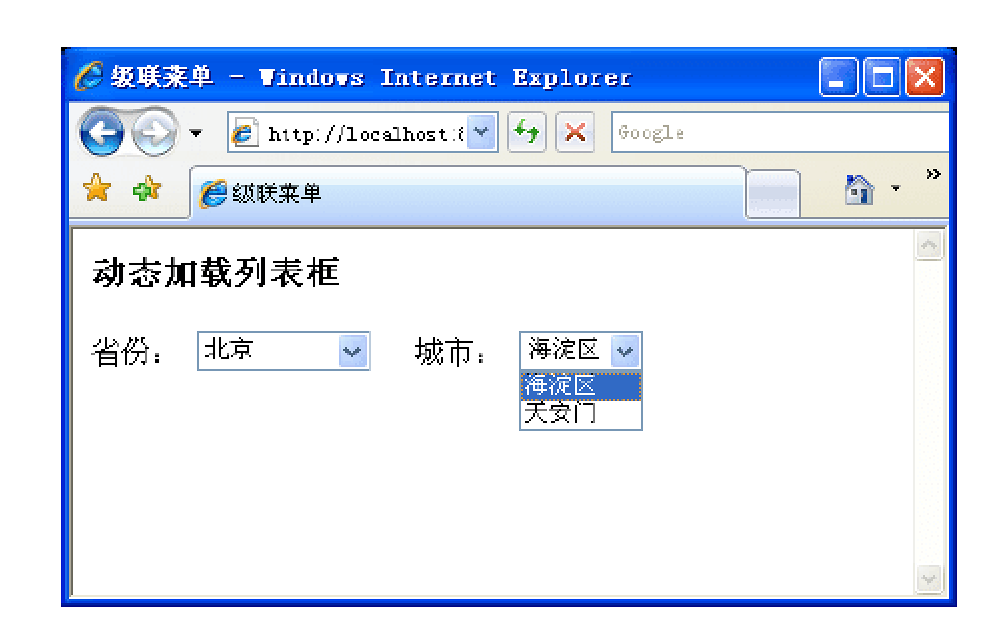


图 13-7　加载后显示的结果

第14章 DOM


内容摘要 | Abstract

DOM 是 HTML 和 XML 文档的编程基础，定义处理执行文档的途径。W3C DOM 提供一套标准的对象用于描述 HTML 和 XML 文档，以及一套标准的接口去访问和处理它们。DOM 还提供一个 API，允许开发人员添加、编辑、移动或删除树中任意位置的节点，从而创建一个应用程序。

本章首先对 DOM 模型进行简要的概述，然后介绍 DOM 结构、DOM 对象中的属性和方法，最后使用 DOM 操作 HTML 和 XML 文档，实现对文档添加、编辑、删除等操作。

学习目标 | Objective

- 了解什么是 DOM 模型
- 理解 DOM 结构模型
- 熟悉 DOM 的接口
- 熟练掌握 DOM 对象的方法和属性
- 掌握 DOM 操作 HTML 文档的使用方法
- 掌握 DOM 操作 XML 文档的使用方法

14.1 DOM 模型概述

DOM（Document Object Model，文档对象模型）是用于操纵 HTML 和 XML 文档的最常用工具之一。DOM 可以看作是一组 API（Application Program Interface，应用编程接口），将 HTML 文档、XML 文档等看作一个文档对象，在接口中存放的是对这些文档对象中数据的存储，并且利用程序对数据进行相应的处理。DOM 实际上是能够让程序和脚本动态访问和更新文档内容、结构和样式的一种语言平台。

DOM 是指 W3C 定义标准的文档对象模型，并以树形结构表示 HTML 和 XML 文档，定义遍历、检查、修改树节点的方法和属性。在实际应用中，DOM 一般被分为不同的部分（核心、XML 和 HTML），分别对应着不同的版本（DOM 1/2/3）。

- **核心 DOM**　用于任何结构化文档的标准模型。
- **XML DOM**　用于 XML 文档的标准模型。
- **HTML DOM**　用于 HTML 文档的标准模型。

DOM 是以层次结构组织的节点或信息片断的集合。这个层次结构允许开发人员在树中导航寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能执行相应的操作。由于 DOM 基于信息层次，因而被认为是基于树或对象。

DOM 规范指定语言绑定所需要实现的接口。简单地说，各种基于 DOM 规范的解析器必须按照 DOM 规范在内存中建立数据，DOM 规范的核心是树模型。对于解析 XML 文档，解析器通过读取 XML 文档并在内存中建立一个树模型。利用 DOM 规范绑定语言编写的程序，就可以对 HTML 或者 XML 文档进行解析。W3C 目前提出 3 个 DOM 规范，分别是 DOM Level 1、DOM Level 2 和 DOM Level 3。

解析器是一个软件应用程序，设计用于分析文档（这里是指 XML 文档）以及做一些特定于该信息的事情。在 SAX 基于事件的 API 中，解析器将向某种监听器发送事件；在 DOM 基于树的 API 中，解析器在内存中建立一个树模型。

使用 W3C DOM 的技术规范可以访问 HTML 和 XML 文档中的数据，并做出相应的处理。和其他访问 HTML 和 XML 文档数据的方法相比较，有如下优势。

- **保证正确的语法和格式良好**

由于 DOM 将 HTML 和 XML 文档在内存中作为节点树来表示，在操作时就不必关心是否出现无结束标记和标记的嵌套等问题，只需要关心节点间的信息就可以了。

- **不受 HTML 和 XML 语法的限制**

由 DOM 创建的节点树是 HTML 和 XML 文档的逻辑表现形式，只显示文档提供的数据和数据之间的关系，并且不受 HTML 和 XML 语法的限制。提取出来的数据可以放到一个文件中，也可以放到一个数据库中，这里只关注数据。

- **简化内部文档的操作**

以树的形式存在，对添加和修改节点这样的操作比传统的文件形式简单得多。

14.2 DOM 结构模型

DOM 模型的核心是树模型，树的结构是 Document 对象，表示整个文档对象，并且仅包含一个子节点。例如对于需要解析的 XML 文档，首先需要使用 DOM 解析器加载文档到内存中，在内存中为 XML 文档建立逻辑形式的树。从本质上说，DOM 就是 XML 文档在内存中的一个结构化视图，它将一个 XML 文档看作是一个节点树，而其中的每一个节点代表一个可以与其进行交互的对象。在这种模式下，XML 文档中的每一个元素都是一个节点，每一个节点都可以包含自己的节点子树，而处于每一个文档顶端的就是根节点。

图 14-1 显示了 XML 文档对象模型（DOM）的类层次结构，其中 W3C 名称用括号括起来，另外还有相关的类名。

从上述结构图中可以看出，其中类 XmlImplementation、XmlNamedNodeMap、XmlNodeList 和 XmlNodeChangedEventArgs 不从 XmlNode 继承。XmlImplementation 类创建 XML 文档，XmlNamedNodeMap 类处理未排序的节点集，XmlNodeList 类处理已排序的节点列表，XmlNodeChangedEventArgs 类处理在 XmlDocument 上注册的事件处理程序。其余的类都继承自 XmlNode 类。



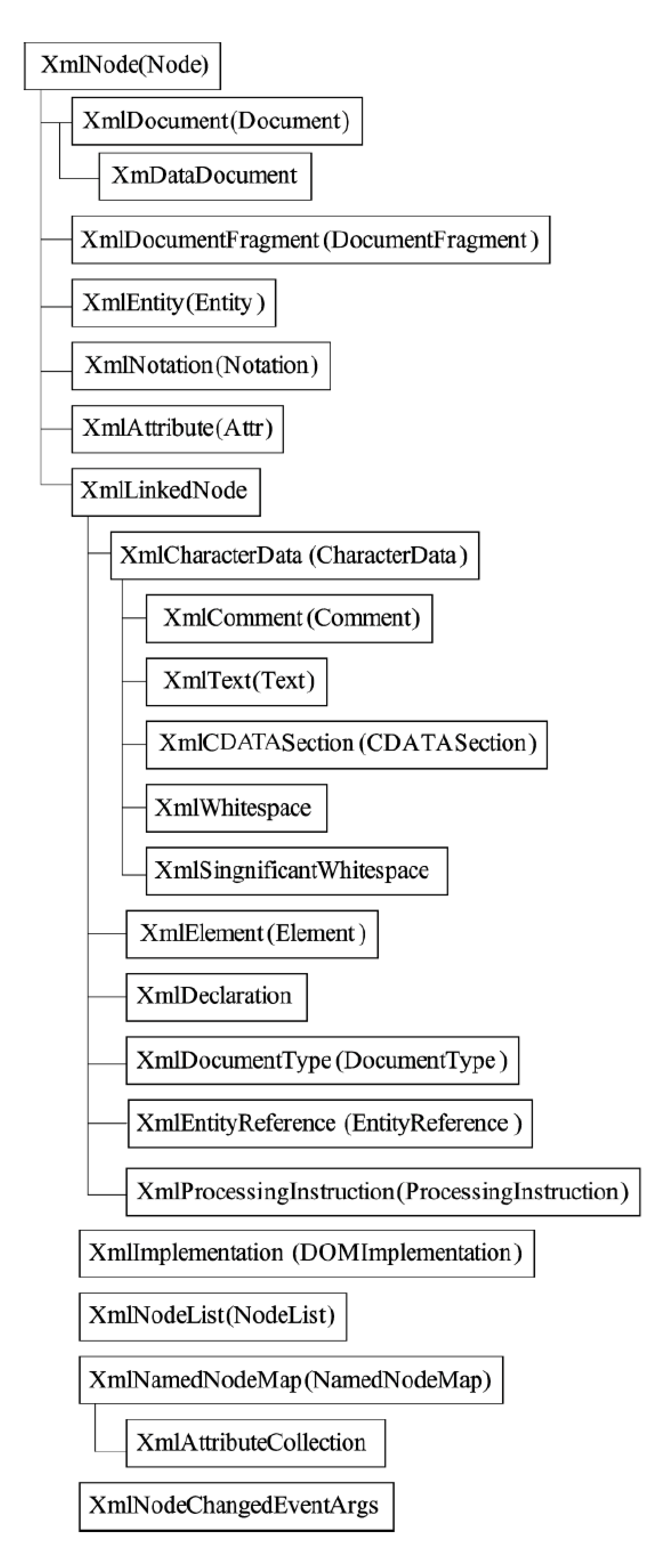


图 14-1　DOM 模型层次结构

14.2.1　DOM 与 HTML

HTML DOM 是针对 HTML 的文档对象模型，定义 HTML 标准对象集合以及访问和操作

HTML 文档的方法。在 HTML DOM 中，HTML 跟 XML 一样是一种树形结构文档，可以作为一个对象的集合来使用。这些对象有属性和方法，同时也可以对事件做出反应，调用对应的函数进行处理。<html>是根（root）节点，<head>、<body>是<html>的子（children）节点，互相之间是兄弟（sibling）节点；<body>下面还包含有子节点<table>、<span>、<p>等，如代码 14.1 所示。

代码 14.1　HTML 文档

```
<html>
<head>
<title>Dom 与 HTML</title>
</head>
<body>
<div>莫失莫忘，很不错，旋律很好听</div>
</body>
</html>
```

在上述代码中，document 是根节点，<html>是根节点；<html>节点下又包括<head>和<body>两个子节点。其中<head>节点包括<title>子节点，<title>节点则包括一段文本，这些文本也是一个节点，称为文本节点；<body>节点中包含一个<div>子节点。在文档中不仅有子节点的关系，还有父节点和兄弟节点的关系。例如<head>节点是<title>节点的父节点，而<head>和<body>节点是兄弟节点。

在 HTML 文档的树形结构中主要包含表示元素、标记的节点和文本的节点。对应的 HTML 的树形结构如图 14-2 所示。

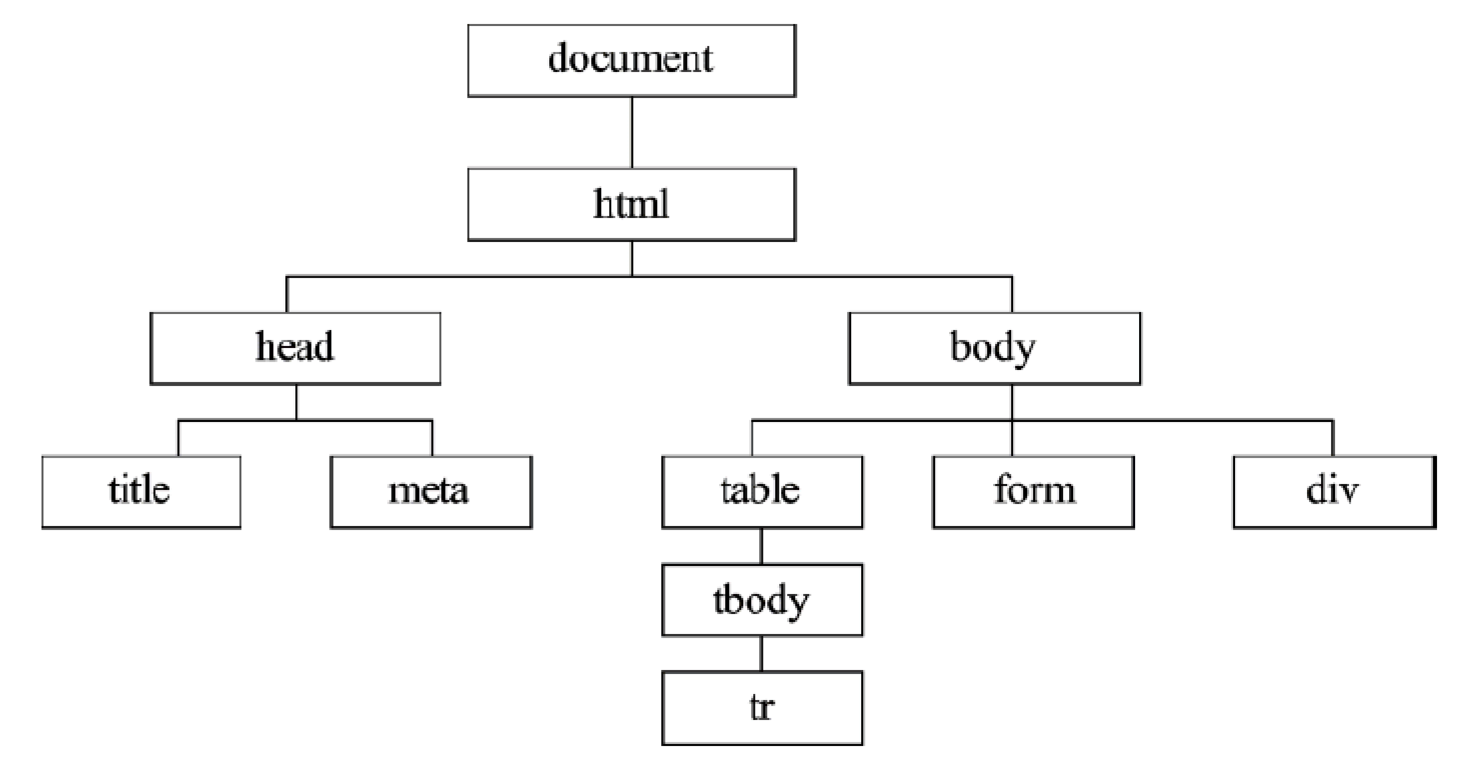


图 14-2　HTML 树形结构图

整个 HTML 文档实际上是由树状层次结构的节点组成的，彼此之间可以通过父节点、子节点和兄弟节点的关系相互引用。当引用一个节点时，实际上就是获取一个节点的对象。通过 DOM 定义的方法，可以对该节点进行操作。

14.2.2　DOM 与 XML

XML DOM 是针对 XML 的文档对象模型。DOM 能够以编程方式读取、处理和修改 XML

文档。XmlReader 类也读取 XML，但只提供非缓存的只进、只读访问。这意味着使用 XmlReader 无法编辑属性值或元素内容，也无法插入和移除节点，而编辑是 DOM 的主要功能。由于 XML 将数据组织为树状，DOM 就是对这棵树的一个对象描述。也就是说解析 XML 文档需要为 XML 文档在逻辑上建立一个树形结构，树的每个节点都是对象，通过存取这些对象就能够存取 XML 文档的内容。

下面创建一个简单的 XML 文档，然后对其进行解析，如代码 14.2 所示。

代码 14.2　XML 文档

```
<?xml version="1.0" encoding="GB2312"?>
<books>
    <book>
        <author>Anne</author>
        <price>25</price>
        <pubdate>05/01/2001</pubdate>
    </book>
    <pubinfo>
        <publisher>MsPress</publisher>
        <state>120 本</state>
    </pubinfo>
</books>
```

上述代码就是一个由节点组成的 DOM 树结构。其中<books>是文档的根节点，在根节点里面包含<book>节点和<pubinfo>节点。而 book 节点又包含 3 个节点：<author>、<price>和<pubdate>，并且都包含相应的数据信息；在<pubinfo>节点中包含两个节点：<publisher>和<state>。在 DOM 中每一个标记都称为一个节点，属性也是一个节点，标记间的数据也是一个节点。对应的 XML 的树形结构如图 14-3 所示。

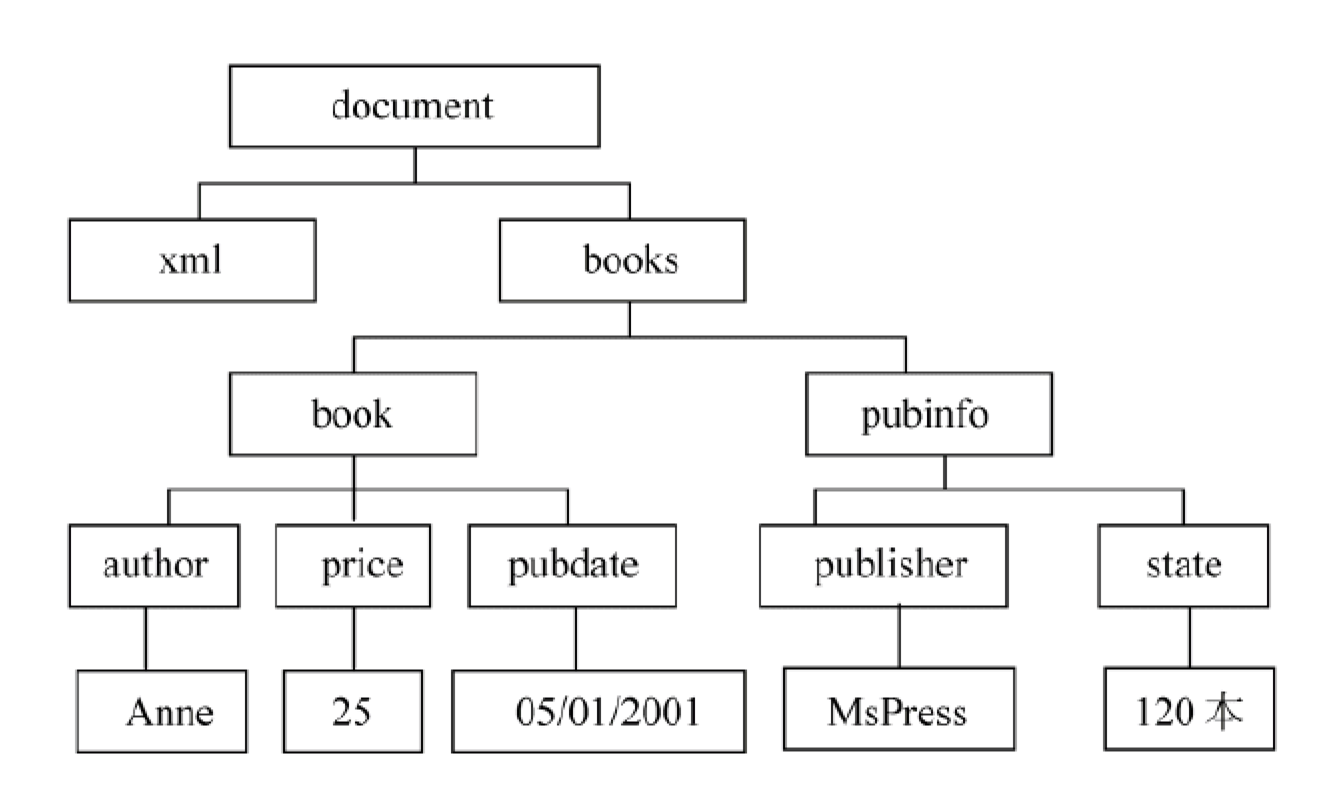
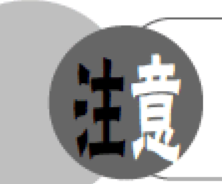


图 14-3　XML 树形结构图

在 XML 文档树形结构图中的每个方框表示一个节点（称为 XmlNode 对象）。XmlNode 对象是 DOM 树中的基本对象。XmlDocument 类（扩展 XmlNode）支持用于对整个文档执行操作（例如，将文档加载到内存中或将 XML 保存到文件中）的方法。此外，XmlDocument

提供查看和处理整个 XML 文档中节点的方法。XmlNode 和 XmlDocument 都具有性能和可用性增强，并通过方法和属性执行下列操作。

- 访问和修改 DOM 特定的节点，例如元素节点、实体引用节点等。
- 除检索节点包含的信息（如元素节点中的文本）外，还检索整个节点。



如果应用程序不需要 DOM 提供的结构或编辑功能，则使用 XmlReader 和 XmlWriter 类提供对 XML 的非缓存的只进流访问。

14.3 DOM 对象

在 DOM 对象中将 XML 文档或 HTML 文档看作一个节点树，那么这个节点树的每一个节点都可以作为对象来看待，例如一个节点、节点中的数据以及节点的属性。由于 DOM 是一组 API 接口，在这个接口里存放着不同的未实例化对象，分别对应着文档中不同类型的节点和数据。

14.3.1 DOM 核心接口

DOM 接口规范包含很多接口，常用接口有 Document 接口、Node 接口、NamedNodeMap 接口、NodeList 接口、Element 接口、Text 接口、CDATASection 接口和 Attr 接口等。其中 Document 接口是对文档进行操作的入口，从 Node 接口继承过来。Node 接口是其他大多数接口的父类，例如 Document、Element、Attr、Text 和 Comment 等接口都继承于 Node 接口。NodeList 接口是一个节点的集合，包含某个节点中的所有子节点。NamedNodeMap 接口也是一个节点的集合，通过该接口可以建立节点名和节点之间的一一映射关系，从而利用节点名可以直接访问特定的节点。

1. Document 接口

Document 接口表示整个 HTML 或 XML 文档。从概念上讲，Document 接口是文档树的根，并提供对文档数据的基本访问。由于元素、文本节点、注释以及处理指令等都不能脱离文档的上下文关系而独立存在，所以 Document 接口提供了创建其他节点对象的方法。通过该方法创建的节点对象都有一个 ownerDocument 属性，用于表明当前节点是由谁所创建以及节点同 Document 之间的关系。Document 接口被实现后会是一个 Document 节点对象，该对象可以包含几个节点，如图 14-4 所示。

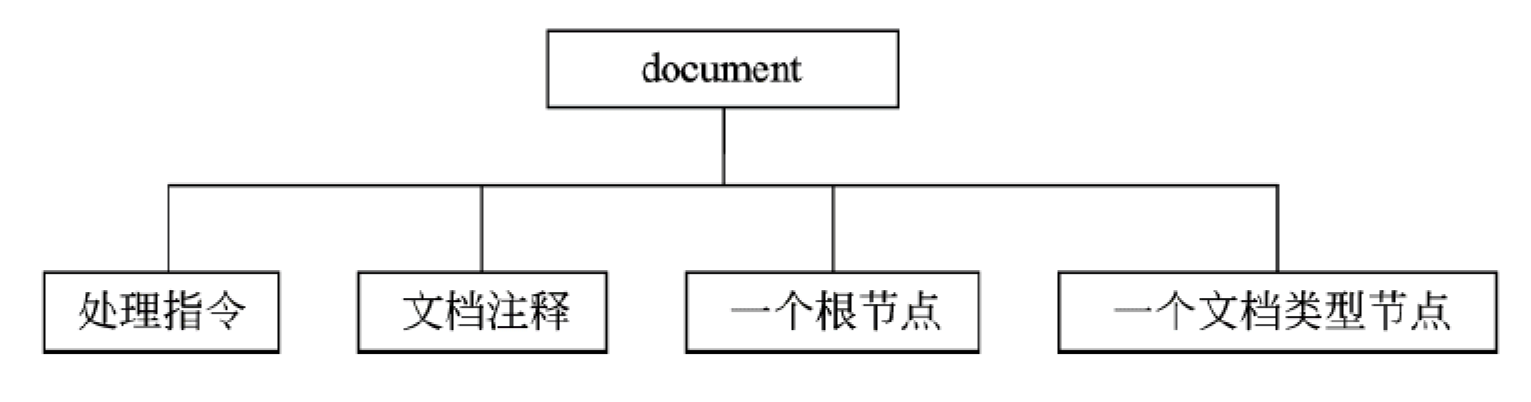


图 14-4　Document 接口示意图

从图中可以看出，Document 节点是 DOM 树中的根节点，即对 HTML 文档进行操作的入口节点。通过 Document 节点可以访问到文档中的其他节点，如处理指令、注释、文档类型以及文档的根元素节点等。另外从图中还可以看出，在 DOM 树中 Document 节点可以包含多个处理指令、多个注释作为其子节点，而文档类型节点和文档根元素节点都是唯一的。

2．Node 接口

Node 接口在整个 DOM 树中非常的重要，DOM 接口中有很大一部分接口继承自 Node 接口，例如 Element、Attr 和 CDATASection 等接口。在 DOM 树中，Node 接口代表树中的一个节点。Node 接口提供访问 DOM 树中元素内容与信息的途径，并提供对 DOM 树中的元素进行遍历的支持。

Node 接口是整个文档对象模型的主要数据类型，表示该文档树中的单个节点。实现 Node 接口的所有对象公开处理子节点的方法时，不是实现 Node 接口的所有对象都有子节点。例如 Text 节点可能没有子节点，且将子节点添加到这样的节点将导致引发 DOMException。

3．NodeList 接口

NodeList 接口提供对节点的有序集合，没有定义或约束如何实现此集合。NodeList 用于表示有顺序关系的一组节点，比如某个节点的子节点序列。另外还出现在一些方法的返回值中，例如 GetNodeByName。在 DOM 中 NodeList 的对象是活动的，换句话说，对文档的改变会直接反映到相关的 NodeList 对象中。如果通过 DOM 获取一个 NodeList 对象，该对象中包含某个 Element 节点的所有子节点的集合，那么当再通过 DOM 对 Element 节点进行操作（添加、删除以及改动节点中的子节点）时，这些改变将会自动地反映到 NodeList 对象中，而不需 DOM 应用程序再做其他额外的操作。

NodeList 中的每个 item 都可以通过一个索引来访问，该索引值从 0 开始。获取一个该接口的实例化对象实际上就是获取一个节点的集合，只不过开始时指针在第一个节点的前面。NodeList 接口被实现后，就是节点集合对象。

4．Element 接口

Element 接口继承自 Node 接口，表示 HTML 或 XML 文档中的一个元素标记，元素可能存在与其相关的属性。由于 Element 接口继承自 Node，所以可以使用一般 Node 接口属性 attributes 来获取元素所有属性的集合。Element 接口上有通过名称获取 Attr 对象或者通过名称获取属性值的方法。在 HTML 中（其中所有属性都有简单的字符串值）可以使用直接访问属性值的方法。

5．NamedNodeMap 接口

NamedNodeMap 接口的对象中包含可以通过名字来访问的一组节点的集合。不过 NamedNodeMap 并不是继承自 NodeList，它所包含的节点在节点集中是无序的。尽管这些节点也可以通过索引进行访问，但这只是提供枚举 NamedNodeMap 中所包含节点的一种简单方法，并不表明在 DOM 规范中为 NamedNodeMap 中的节点规定一种排列顺序。NamedNodeMap 表示一组节点和其唯一名字的一一对应关系，这个接口主要用于表示属性节点。与 NodeList

相同，在 DOM 中 NamedNodeMap 对象是 live，即活动的。

6. Text 接口

Text 接口继承自 CharacterData，并且表示 Element 或者 Attr 的文本内容（在 XML 中称为字符数据）。如果元素的内容中没有标记，则文本包含在实现 Text 接口的单个对象中，此接口是该元素的唯一子节点。如果有标记，则将其解析为信息项（元素、注释等等）和组成子元素列表的 Text 节点。首先通过 DOM 使文档可用时，文本的每个块只有一个 Text 节点。用户可以创建表示给定元素内容相邻的 Text 节点，没有任何插入标记，但应该知道无法在 XML 或 HTML 中表示这些节点之间的分隔，因此它们通常不会保持在 DOM 编辑会话之间。Node.normalize()方法为每个文本块将所有这样的相邻 Text 对象合并为单个的节点。不对 Text 节点的内容进行任何词汇检查，并且根据节点在文档中的位置，有些字符必须在使用字符引用进行序列化期间转义。例如，当文本内容是元素或属性的一部分字符“<&”，当文本内容是元素一部分的字符序列“]]>”，当文本内容是属性一部分的引号字符“"”。

14.3.2 DOM 基本对象

当用编程语言实现一个接口的对象时，就可以称该对象为 DOM 对象。例如在 Atrr 这个接口中，里面封装关于节点属性的操作方法，例如获取属性的名称、属性的值等。如果一个 Attr 对象实现这个接口，那么这个对象就是 DOM 对象，即属性操作对象。

以 XML DOM 为例，在 DOM 对象中主要有下面几个常用的对象。

- **文档对象**

XML 文档既是一种对象，同时又代表整个 XML 文档，由根元素和子元素组成。该对象由 Document 对象实现。如果对一个 XML 文档进行操作，首先要获取整个文档对象，然后再根据程序的需要调用该对象的其他子对象。

- **节点对象**

XML 节点对象代表 XML 文档内部的单个节点。由 Node 接口实现的对象，利用该对象可以完成对节点追加等方面的操作。

- **节点列表**

XML 文档模块列表代表节点的集合。由 NodeList 接口实现的对象，通过该对象可以对节点进行大批量的读取和操作。

实现一个接口的对象之后，就可以依据该对象对 XML 文档的节点树中相关对象进行操作。可以这样理解，XML 文档的节点树是一个实体，要对这个实体对象操作需要相关的方法，而接口中存在这些方法，对象是一个接口的实现。

1. Document

DOM 树的根节点是个 Document 对象，该对象的 documentElement 属性应用表示文档根元素的 Element 对象（对于 HTML 文档，这个就是<html>标记）。JavaScript 操作 HTML 文档时 Document 指向整个文档，<body>和<table>等节点类型即为 Element。Comment 类型的节点则是指文档的注释。

Document 定义的方法大多数是生产型方法，主要用于创建可以插入文档中的各种类型的节点。常用的 Document 方法如表 14-1 所示。

表 14-1　Document 的方法

方法	说明
createAttribute()	用指定的名字创建新的 Attr 节点
createComment()	用指定的字符串创建新的 Comment 节点
createElement()	用指定的标记名创建新的 Element 节点
createTextNode()	用指定的文本创建新的 TextNode 节点
getElementById()	返回文档中具有指定 id 属性的 Element 节点
getElementsByTagName()	返回文档中具有指定标记名的所有 Element 节点

2. Element

对于 Element 节点，可以通过调用 getAttribute()、setAttribute()和 removeAttribute()方法来查询、设置或者删除一个 Element 节点，例如<table>标记的 border 属性。Element 常用的属性是 tagName，使用该属性可以获取元素标记名称，例如<p>元素为 P。HTML 文档返回的 tagName 均为大写。Element 常用的方法如表 14-2 所示。

表 14-2　Element 的方法

方法	说明
getAttribute()	以字符串形式返回指定属性的值
getAttributeNode()	以 Attr 节点的形式返回指定属性的值
getElementsByTagName()	返回一个 Node 数组，包含具有指定标记名的所有 Element 节点的子节点，其顺序为在文档中出现的顺序
hasAttribute()	如果元素具有指定名字的属性，则返回 true
removeAttribute()	从元素中删除指定的属性
removeAttributeNode()	从元素的属性列表中删除指定的 Attr 节点
setAttribute()	将指定的属性设置为指定的字符串值，如果该属性不存在则添加一个新属性
setAttributeNode()	将指定的 Attr 节点添加到该元素的属性列表中

Attr 对象代表文档元素的属性，有 name 和 value 等属性，可以通过 Node 接口的 attributes 属性或者调用 Element 接口的 getAttributeNode()方法来获取。不过在大多数情况下，使用 Element 元素属性的最简单方法是 getAttribute()和 setAttribute()两种方法，而不是 Attr 对象。

3. Node

Node 对象定义一系列属性和方法，非常方便地遍历整个文档。用 parentNode 属性和 childNodes[]数组可以在文档树中上下移动。通过遍历 childNodes[]数组或者使用 firstChild 和 nextSibling 属性进行循环操作，也可以使用 lastChild 和 previousSibling 进行逆向循环操作，还可以枚举指定节点的子节点。而调用 appendChild()、insertBefore()、removeChild()以及 replaceChild()方法可以改变一个节点的子节点从而改变文档树。

childNodes[]的值实际上是一个 NodeList 对象。因此可以通过遍历 childNodes[]数组的每个元素获取给定节点的所有子节点。通过递归可以获取树中的所有节点。Node 对象的常用属

性如表 14-3 所示。

表 14-3　Node 对象的属性

属性	说明
attributes	如果该节点是一个 Element，则以 NamedNodeMap 形式返回该元素的属性
childNodes	以 Node 的形式存放当前节点的子节点，如果没有子节点，则返回空数组
firstChild	以 Node 的形式返回当前节点的第一个子节点，如果没有子节点，则为 null
lastChild	以 Node 的形式返回当前节点的最后一个子节点，如果没有子节点，则为 null
nextSibling	以 Node 的形式返回当前节点的下一个节点，如果没有这样的节点，则返回 null
nodeName	节点的名字，Element 节点则代表 Element 的标记名称
nodeType	代表节点的类型
parentNode	以 Node 的形式返回当前节点的父节点，如果没有父节点，则为 null
previousSibling	以 Node 的形式返回紧挨当前节点、位于它之前的兄弟节点，如果没有这样的节点，则返回 null

Node 对象的常用方法如表 14-4 所示。

表 14-4　Node 对象的方法

方法	说明
appendChild()	通过将一个节点增加到当前节点的 childNodes[]组，给文档树增加节点
cloneNode()	复制当前节点，或者复制当前节点以及它的所有子节点
hasChildNodes()	如果当前节点拥有子节点，则将返回 true
insertBefore()	给文档树插入一个节点，位置在当前节点的指定子节点之前。如果该节点已经存在，则删除后再插入到它的位置
removeChild()	从文档树中删除并返回指定的子节点
replaceChild()	从文档树中删除并返回指定的子节点，用另一个节点替换

14.3.3　创建 DOM 对象

如果对文档进行操作，那么必须使用相应的编程语言实现 DOM 接口，创建一个 DOM 对象。创建 DOM 文档对象的语法如下所示：

```
var xmlDoc=new ActiveXObject("Msxml2.DOMDocument.4.0");
```

对象创建好以后就可以对文档进行操作，例如将一个 XML 文档加载到内存中的节点树进行添加节点、遍历节点以及修改节点等。

不同的语言创建 DOM 对象的方法也不同，下面列出几种不同语言创建 DOM 对象的语法，如代码 14.3 所示。

代码 14.3　创建 DOM 对象的几种不同方法

```
VBScript
dim xmldoc
set xmldoc = CreateObject("Microsoft.XMLDOM")
JavaScript
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
```

```
xmlDoc.load("student.xml");
ASP
<%
  DIM oXMLDOM
  set oXMLDOM = server.CreateObject("Microsoft.XMLDOM")
  oXMLDOM.load("student.xml")
%>
php
<?php
  $dom = xmldocfile("student.xml") or die("what emp")
  $root = $dom->root();
?>
JAVA
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document document = builder.pares(new File("student.xml"));
```

在上述代码中，都是首先创建一个对象，然后通过该对象作为入口对 XML 文档的节点树对象进行操作。

14.4　使用 DOM 操作 HTML 文档

下面通过示例说明如何使用 DOM 操作 HTML 文档，例如遍历文档中的节点、访问指定的节点、对文档进行添加以及修改和删除操作等。

14.4.1　遍历文档的节点

在 DOM 中，HTML 文档各个节点被视为各种类型的 Node 对象。每个 Node 对象都有自己的属性和方法，利用这些属性和方法可以遍历整个文档树。由于 HTML 文档的复杂性，DOM 定义 nodeType 来表示节点的类型。

在使用 DOM 时必须待文档被装载完毕后再执行遍历等行为操作文档，具体实现如代码 14.4 所示。

代码 14.4　loop.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312">
<title>读取 html 文档中的所有元素</title>
<script language="javascript" type="text/javascript">
var elementName = "";                //全局变量，保存 Element 标记名
function countTotalElement(node) {  //参数 node 是一个 Node 对象
var num = 0;
```

```
if(node.nodeType == 1) {          //检查 node 是否为 Element 对象
    num++;                        //如果是，计数器加 1
    elementName = elementName + node.tagName + "\r\n"; //保存标记名
}
var childrens = node.childNodes;                  //获取 node 的全部子节点
for(var i=0;i<childrens.length;i++) {
    num += countTotalElement(childrens[i]);      //在每个子节点上进行递归操作
}
return num;
}
</script>
</head>
<body>
<a href="#" onclick="alert('标记总数: ' + countTotalElement(document) + '\r\n
全部标记如下: \r\n' +elementName);elementName='';">开始统计标记</a>
</body>
</html>
```

该示例使用 childNodes[]和递归方式来遍历整个文档，统计文档中出现的 Element 标记总数，并将 Element 标记名称全部打印出来。将上述代码保存为 loop.html，在浏览器地址栏中输入“http://localhost:8080/loop.html”查看页面运行效果。当单击【开始统计标记】超级链接按钮时，则会弹出一个对话框，并将 html 文档中的元素个数和元素标记名称显示出来，运行结果如图 14-5 所示。



图 14-5 运行结果

14.4.2 搜索特定元素

在使用 DOM 的过程中，有时需要定位到文档中的某个特定节点，或者具有特定类型的节点列表。这种情况下可以调用 Document 对象的 getElementsByTagName()和 getElementById()方法来实现。

document.getElementsByTagName()返回文档中具有指定标记名的全部 Element 节点数组（也是 NodeList 类型）。Element 在数组中出现的顺序就是在文档中出现的顺序。传递给

getElementsByTagName()的参数忽略大小写。例如定位到第一个<table>标记，则编写如下代码：

```
document.getElementsByTagName("table")[0];
```

如果需要定位到<body>标记，则可以使用 document.body，因为它是唯一的。

getElementsByTagName()返回的数组取决于文档。一旦文档改变，返回结果也立即改变。相比而言 getElementById()则比较灵活，可以随时定位到目标，只是要实现给目标元素一个唯一的 id 属性值。在这里不再举例说明，可以参照第 13 章中“级联菜单”示例。

14.4.3　修改内容

本节的示例采用数组缓存的方式修改 HTML 文档中的内容，并且演示如何使用 DOM 创建表格单元行，具体如代码 14.5 所示。

代码 14.5　update.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312">
<title>修改文档中的内容</title>
<script language="javascript">
function reverseTable() {
var node = document.getElementsByTagName("table")[0];  //第一个表格
var child = node.getElementsByTagName("tr");        //取得表格内的所有行
var newChild = new Array();                         //定义缓存数组，保存行内容
for(var i=0;i<child.length;i++) {
    newChild[i] = child[i].firstChild.innerHTML;
}
node.removeChild(node.childNodes[0]);               //删除全部单元行
var header = node.createTHead();                    //新建表格行头
for(var i=0;i<newChild.length;i++) {
    //插入一个单元行
    var headerrow = header.insertRow(i);
    //在单元行中插入一个单元格
    var cell = headerrow.insertCell(0);
    //在单元格中创建 TextNode 节点
    cell.appendChild(document.createTextNode(newChild[newChild.length-
    i-1]));
}
}
</script>
</head>
<body>
<table width="200" border="1" cellpadding="4" cellspacing="0">
      <tr>
```

```
            <td height="25">莫失莫忘</td>
        </tr>
        <tr>
            <td height="25">杨柳</td>
        </tr>
        <tr>
            <td height="25">乱红</td>
        </tr>
        <tr>
            <td height="25">胃疼的鱼</td>
        </tr>
</table>
<br>
<input type="button" name="reverse" value="重新排序" onclick="reverseTable()">
</body>
</html>
```

在 HTML 文档中，布局和定位常常通过表格<table>来实现。该示例演示将表格中的四个单元行重新排序。如果没有使用<tbody>标签，则<table>将全部的<tr>当作是属于一个子节点<tbody>。在浏览器地址栏输入“http://localhost:8080/update.html”，当页面初始化之后运行结果如图 14-6 所示。单击【重新排序】按钮之后，页面的运行结果如图 14-7 所示。

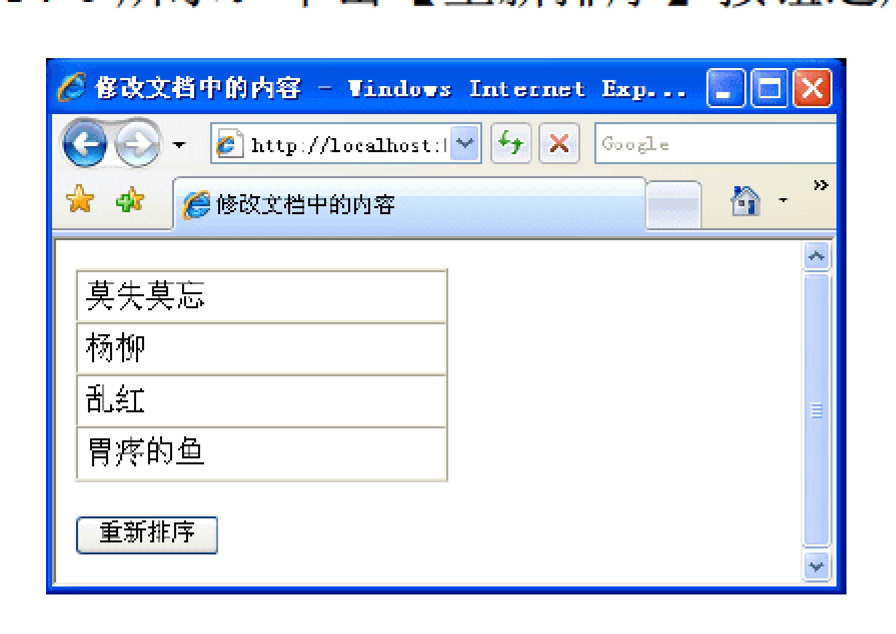


图 14-6　页面初始化

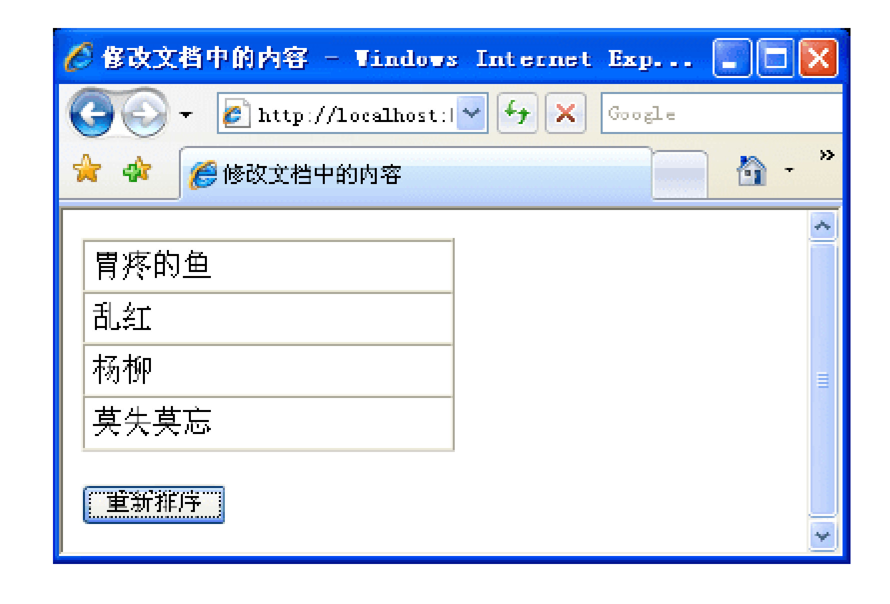


图 14-7　单击按钮之后

14.4.4　添加和删除内容

在掌握遍历、搜索和修改文档之后，接下来讲解使用 Node 的属性和方法向文档中添加和删除内容。删除节点需要用到 Node 对象的 removeChild()方法，该方法表示从文档树中删除并返回指定的子节点。当向文档中添加内容时，则会产生一个【删除】按钮。首先定义一个添加函数，具体如代码 14.6 所示。

代码 14.6　insertStr()函数

```
<script language="javascript" type="text/javascript">
function insertStr()
{
```

```
    //获取文本框中输入的值
    var name = document.getElementById("str").value;
    //新建一个 TextNode 节点
    var test = document.createTextNode(name);
    //新建一个 td 类型的 Element 节点
    var td = document.createElement("td");
    //新建一个 tr 类型的 Element 节点
    var tr = document.createElement("tr");
    tr.setAttribute("id",name);
    td.appendChild(test);//将节点 test 加入 td 中
    tr.appendChild(td);  //将节点 td 加入 tr 中
    //创建元素删除按钮
     var deleteButton = document.createElement("input");
     deleteButton.setAttribute("type", "button");
     deleteButton.setAttribute("value", "删除");
     deleteButton.onclick = function (){ deleteSort(name); };
     var td = document.createElement("td");
     td.appendChild(deleteButton);
     tr.appendChild(td);
     //添加 deleteButton 元素
     document.getElementById("sortList").appendChild(tr);
  }
  </script>
```

从以上代码中可以看出，在 insertStr()函数中通过 id 属性值为 str 的元素在文档中添加一行，然后用 appendChild()方法向行中依次添加每个子元素。并且创建一个【删除】按钮，当单击【删除】按钮之后调用删除函数将添加的内容删除。接下来定义删除函数，具体如代码 14.7 所示。

代码 14.7　deleteSort(id)函数

```
  // 删除成员函数
        function deleteSort(id) {
          if (id!=null){
              var obj=document.getElementById(id);
              var sortList= document.getElementById("sortList");
              sortList.removeChild(obj);
          }
          }
```

该代码通过 id 属性（id 是添加行的关键字）和 removeChild()方法来完成删除功能。最后编写页面布局代码，如代码 14.8 所示。

代码 14.8　add.html

```
  <form name="form1" method="post" action="#">
```

```
        <input name="str" type="text" id="str" value="" />
        <input name="insert" type="button" id="insert" value="留言"
        onclick="insertStr()"/>
    </form>
    <h3>网友留言列表: </h3>
    <table width="100" border="1" cellspacing="0" cellpadding="0" id="table1">
    <tbody id="sortList">
    </tbody>
    </table>
```

此时程序已经设置完成，接下来在浏览器中查看运行效果。在浏览器地址栏中输入“http://localhost:8080/add.html”，当用户在文本框中输入内容之后，单击【留言】按钮则会在页面中创建一个表格显示留言信息，并且创建一个删除按钮。单击【删除】按钮可以将留言删除。运行结果如图 14-8 所示。

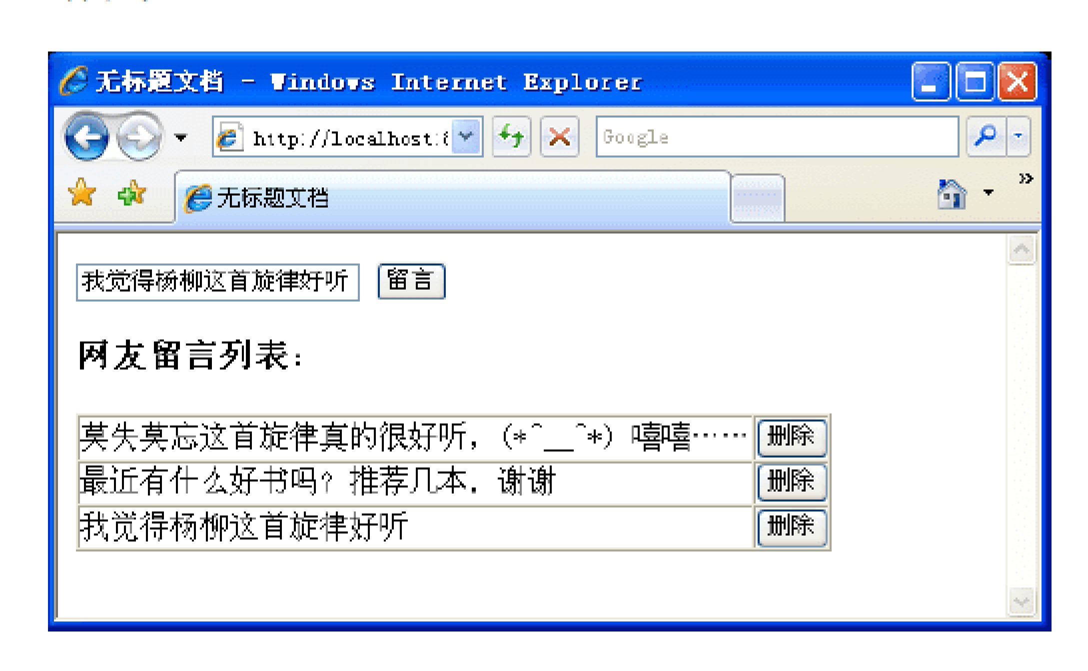


图 14-8　运行结果

14.5　使用 DOM 操作 XML 文档

在数据表示方面，XML 文档更加结构化。DOM 在支持 HTML 的基础上提供了一系列的 API，支持针对 XML 的访问和操作。利用这些 API 可以从 XML 中提取信息，动态地创建这些信息的 HTML 呈现文档。下面通过示例演示如何使用 DOM 遍历、复制、修改和删除 XML 文档的节点。

14.5.1　创建 XML 文档

创建 XML 文档首先需要创建一个 DOM 对象和 XML 声明，接着使用 createNode()方法定义节点，并使用 createElement()方法创建元素。节点和元素创建好以后可以使用 createAttribute()方法添加属性，并使用 setAttributeNode()方法将属性值添加给节点 name。接下来使用 appendChild()方法定义节点和子节点。最后保存 XML 文档，具体实现如代码 14.9 所示。

代码 14.9　createxml.html

```
    <html>
```

```
<head>
 <title>创建 XML 文档</title>
</head>
<script type="text/javascript">
 var doc = new ActiveXObject("Msxml2.DOMDocument");
 //创建 XML 声明
 var p = doc.createProcessingInstruction("xml","version='1.0'  encoding=
 'GB2312'");
 //添加 XML 声明
 doc.appendChild(p);
 //创建节点 students
 var root = doc.createNode(1,"students","");
 //创建节点 student
 var n = doc.createNode(1,"student","");
 //创建元素
 var name = doc.createElement("name");
 var address = doc.createElement("address");
 var tele = doc.createElement("tele");
 var email = doc.createElement("email");
 //指定文本
 name.text ="Anne Lee";
 address.text ="London";
 tele.text="15993705626";
 email.text ="jiangnan@yahoo.com.cn";
 //创建属性
 var id = doc.createAttribute("id");
 id.value ="E001";
 //将属性添加给节点 name
 name.setAttributeNode(id);
 //添加子节点
 n.appendChild(name);
 n.appendChild(address);
 n.appendChild(tele);
 n.appendChild(email);
 //为 root 添加节点
 root.appendChild(n);
 //将创建的 students 节点添加到 Document 对象中
 doc.appendChild(root);
 //客户端用 FSO 保存 XML
var fso = new ActiveXObject("Scripting.FileSystemObject");
var newFileObject = fso.CreateTextFile("c:/student.xml", true);
newFileObject.WriteLine(doc.xml);
newFileObject.close();
 </script>
<body onload="javaScript:alert('XML 文档已经创建！');"></body>
</html>
```

将上述代码保存为 createxml.html，在浏览器地址栏中输入“http:localhost:8080/createxml.html”打开页面之后，则触发 onload 事件，弹出一个对话框提示“XML 文档已经创建！”，然后打开 C 盘，可以在根目录下找到创建的 XML 文档。双击打开 XML 文档，页面运行效果如图 14-9 所示。

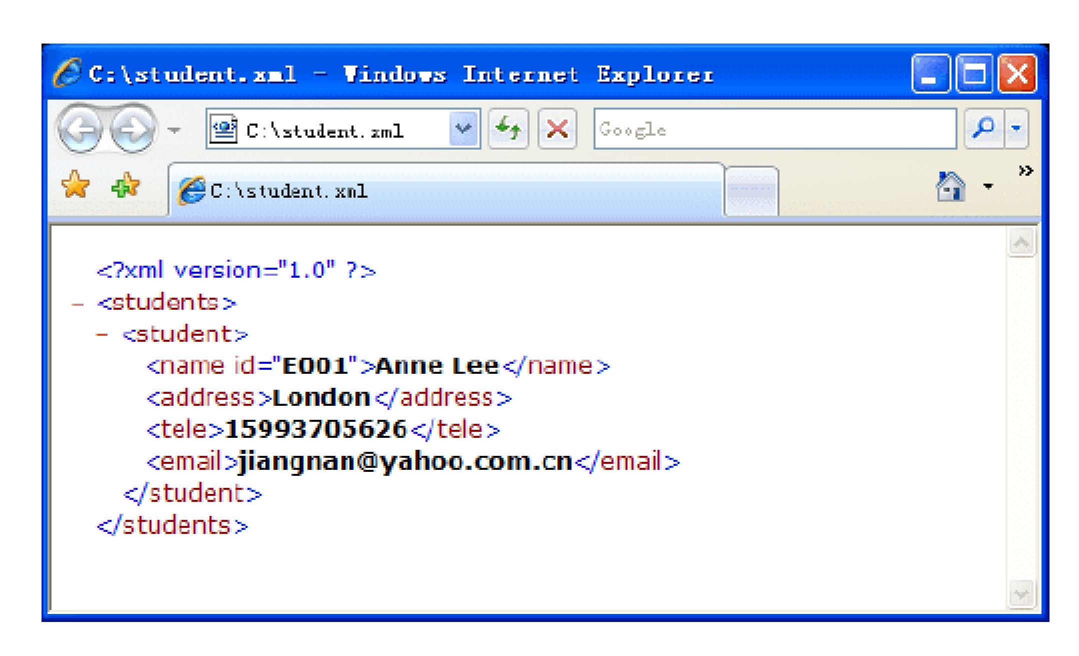
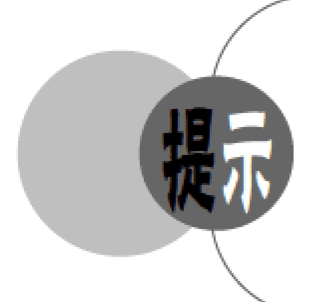


图 14-9　student.xml 文档



出于对网络安全的考虑，save()方法需要在服务器端才可以保存 XML 文件。客户端要想保存，需要使用 FSO 对象，并且需要注册 scrrun.dll，以及将 Internet Explorer 的【工具】|【Internet 选项】|【安全】|【自定义级别】中启用【对未标记为可安全执行脚本的 ActiveX 控件初始化并执行脚本】复选框。

14.5.2　遍历 XML 文档

XML 文档创建好以后就可以使用 DOM 对象中的一些方法和属性对 XML 文档进行遍历，下面通过示例说明如何将 XML 文档中的节点和值读取到 HTML 页面。首先需要将创建好的 XML 文档使用 DOM 对象添加到页面中，并获取 XML 文档中根节点元素，然后根据 DOM 对象的 nodeName 和 nodeValue 属性获取 XML 文档中的节点元素和节点的值，具体实现如代码 14.10 所示。

代码 14.10　遍历 XML 文档

```
<html>
<head>
 <title>操作 XML 文档</title>
</head>
<script type="text/javascript">
 var doc = new ActiveXObject("Microsoft.XMLDOM");
 doc.async="false";
 //导入 XML 文档
 doc.load("c:/student.xml");
var x = doc.documentElement.childNodes;
for(var i=0;i<x.length;i++)
 {
   //获取根节点
```

```
        document.write(x[i].nodeName);
        document.write("<br />");
        //获取子节点
        for(var j=0;j<x[i].childNodes.length;j++)
        {
            document.write(x[i].childNodes[j].nodeName);
            document.write(": ");
            //获取值
            document.write(x[i].childNodes[j].firstChild.nodeValue);
            document.write("<br />");
        }
      }
    </script>
    <body></body>
    </html>
```

此时代码已经编写完成，在浏览器中查看运行效果。在浏览器地址栏中输入“http://localhost:8080/bianli.html”之后，在页面中可以看到 XML 文档的根节点和子节点，并且还显示子节点的属性值。页面运行效果如图 14-10 所示。



14-10　遍历 XML 文档

14.5.3　复制和修改节点

下面通过示例讲解如何复制 XML 文档中的节点，并且修改节点的属性和值。复制节点使用 cloneNode()方法，修改元素属性使用 setAttribute()方法，最后修改复制节点的值，具体实现如代码 14.11 所示。

代码 14.11　copy.html

```
    <html>
    <head>
     <title>操作 XML 文档</title>
    </head>
    <script type="text/javascript">
     var doc = new ActiveXObject("Microsoft.XMLDOM");
     doc.async="false";
     //导入 XML 文档
     doc.load("c:/student.xml");
```

```
    //获取所有根节点
    var stu = doc.getElementsByTagName('student')[0];
    //复制节点
    var m = stu.cloneNode(true);
    //添加到 XML 文档中
    doc.documentElement.appendChild(m);
    //修改复制节点的属性
    var x=doc.getElementsByTagName('name');
    x[1].setAttribute("id","E002");
    //修改复制节点的值
    var y = doc.getElementsByTagName("name")[1].childNodes[0];
    y.nodeValue="JoJo";
    var y1 = doc.getElementsByTagName("address")[1].childNodes[0];
    y1.nodeValue="beijing";
    var y2 = doc.getElementsByTagName("tele")[1].childNodes[0];
    y2.nodeValue="13782671736";
    var y3 = doc.getElementsByTagName("email")[1].childNodes[0];
    y3.nodeValue="iceh@126.com";
    //客户端用 FSO 保存 XML
   var fso = new ActiveXObject("Scripting.FileSystemObject");
   var newFileObject = fso.CreateTextFile("c:/student.xml", true);
   newFileObject.WriteLine(doc.xml);
   newFileObject.close();
   </script>
   <body></body>
   </html>
```

在浏览器中运行上述代码之后，XML 文档中的内容被修改。然后在 C 盘的根目录下找到 XML 文档，双击打开查看效果。XML 文档运行结果如图 14-11 所示。

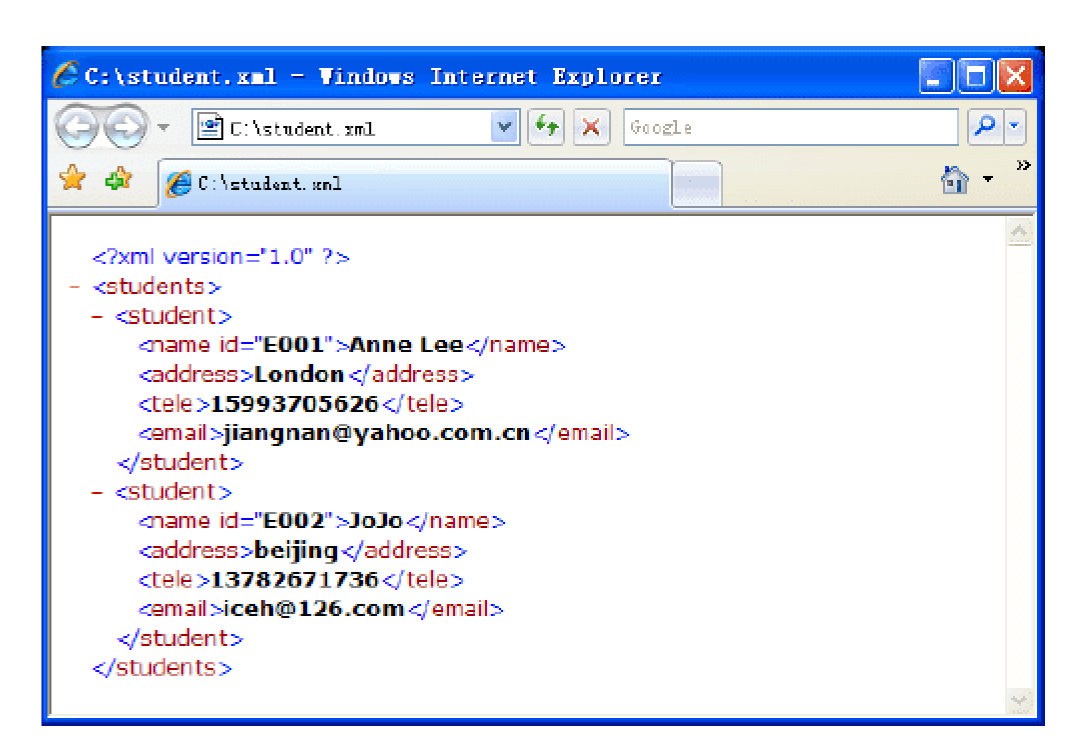


图 14-11　运行结果

14.5.4　删除节点

在上面示例中讲解了创建 XML 文档、遍历 XML 文档、复制和修改 XML 的节点，下面接着讲解使用 DOM 对象的一些属性和方法删除 XML 文档中的节点。当定位需要删除的节点

时可以通过使用 parentNode 属性和 removeChild()方法删除此节点，当一个节点被删除时其所有子节点也会被删除。nodeValue 属性可用于改变或清空文本节点的值，具体实现如代码 14.12 所示。

代码 14.12　delete.html

```
<html>
<head>
 <title>操作 XML 文档</title>
</head>
<script type="text/javascript">
 var doc = new ActiveXObject("Microsoft.XMLDOM");
 doc.async="false";
 //导入 XML 文档
 doc.load("c:/student.xml");
 //删除第一个 name 元素
 var x = doc.getElementsByTagName('name')[0];
 x.parentNode.removeChild(x);
 //清空文本节点
 var z =doc.getElementsByTagName("tele")[0].childNodes[0];
 z.nodeValue="";
 //删除第二个 student 节点
 var y = doc.getElementsByTagName('student')[1];
 doc.documentElement.removeChild(y);
 //客户端用 FSO 保存 XML
var fso = new ActiveXObject("Scripting.FileSystemObject");
var newFileObject = fso.CreateTextFile("c:/student.xml", true);
newFileObject.WriteLine(doc.xml);
newFileObject.close();
</script>
<body></body>
</html>
```

将上述代码保存为 delete.html，在浏览器地址栏中输入“http://localhost:8080/delete.html”之后，初始化该页面。在 C 盘根目录下找到 XML 文档，双击打开，页面效果如图 14-12 所示。

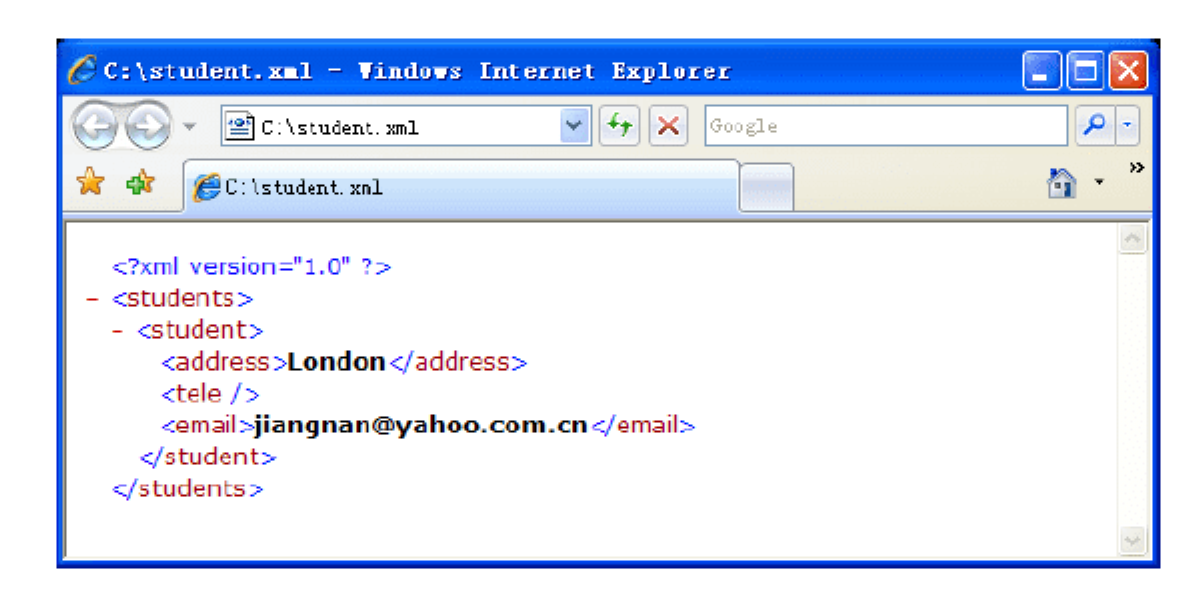
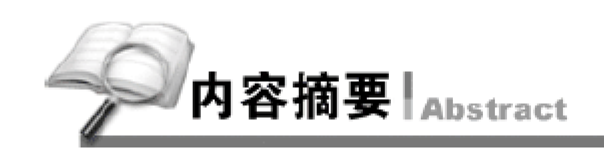


图 14-12　删除节点

第 4 篇　PHP+Ajax 组合篇

第15章 Ajax 客户端应用



通过对前面章节的学习，了解到 Ajax 是一种 Web 应用程序客户端技术。Ajax 结合了脚本语言 JavaScript、层叠样式表 CSS、超文本标记语言 HTML、异步传输对象 XMLHttpRequest 和文档对象模型 DOM 等多种技术。运行在浏览器上的 Ajax 应用程序以一种异步的方式与 Web 服务器通信，处理响应过程实现无刷新更新页面元素。通过使用 Ajax 技术，可以提供丰富的、基于浏览器的用户体验。

本章主要讲解如何使用 Ajax 的核心技术在客户端与服务器端进行通信，即发送请求和处理响应，并通过示例说明 XMLHttpRequest 对象如何向服务器发送请求，以及使用 JavaScript 处理服务器响应。



- 了解基本的请求和处理方式
- 理解如何使用 GET 和 POST 方法发送请求
- 掌握如何发送 XML 格式请求
- 掌握如何发送 JSON 格式请求
- 掌握服务器处理普通响应的方法
- 掌握服务器处理 XML 的方法
- 熟悉使用 Ajax 的开发过程

15.1 在 HTTP 请求中包含参数

在传统的 JavaScript 编程中，如果从服务器上的文件或数据库中获取信息，或者向服务器发送信息，就必须利用一个 HTML 表单向服务器发送数据。而用户则需要单击【提交】按钮来发送或获取信息，并等待服务器的响应，然后在一个新的页面中加载结果。

由于当用户提交输入信息之后，服务器都会返回一个新的页面，使得传统的 Web 应用程序运行得非常缓慢。通过 Ajax 技术，JavaScript 会通过使用其 XMLHttpRequest 对象直接与服务器进行通信。

使用 XMLHttpRequest 对象向服务器发送请求非常简单，需要做的事情就是为其传递一个服务器页面的 url，这个页面将产生数据。使用 HTTP 请求，Web 页面可以向服务器进行请求

并得到来自服务器的响应，而不是加载页面。用户可以停留在同一个页面，并且不会注意到脚本在后台发送请求，或向服务器发送数据。

15.1.1 发送包含参数的普通请求

Ajax 运行机制的核心是在客户端使用 XMLHttpRequest 对象发送异步请求，服务器端使用特定的语言对请求进行处理并返回，然后将信息显示在页面上。使用 Ajax 首先要做的就是发送 HTTP 请求，这主要包括普通请求和带格式的 XML 请求。

一般情况下向服务器发送一个请求，如果这个请求没有任何请求参数则其没有任何意义。因为如果没有请求参数，服务器就得不到上下文数据，也无法根据上下文数据为客户端创建不同的响应，服务器就会向每一个客户端发送同样的响应。

想要充分发挥 Ajax 技术的强大功能，就需要向服务器发送一些带参数的请求。例如有一个用户注册表单，其中包含验证输入用户名的功能，程序需要根据用户输入的用户名判断是否与已存在用户名冲突。这就需要在用户输入用户名后以某种方式将该值发送给服务器，服务器端再访问数据库进行判断并返回相应结果。

XMLHttpRequest 对象的工作与以往惯用的 HTTP 技术（GET 和 POST）一样。GET 方法就是将值作为“名称/值对”放在请求的 url 中传递。资源 url 中有一个问号“?”，问号后面就是“名称/值对”，即采用 name=value 的形式，各个“名称/值对”之间使用“&”分隔。使用 POST 方法向服务器发送参数时，与 GET 方法几乎一样。POST 方法会将参数编码为“名称/值对”，形式为 name=value，每个“名称/值对”之间也是使用“&”分隔。这两种方法的主要区别在于 POST 方法将参数串放在请求中发送，GET 方法将参数追加到 url 中发送。

每个方法都有其各自的优势。由于 GET 请求的参数编码到请求 url 中，所以可以在浏览器中为该 url 建立收藏，之后可以很容易地重新发送请求。不过如果是异步请求就没有此功能。从发送到服务器的数据量来讲，POST 方法更为灵活。使用 GET 请求发送的通常是固定数据量，因浏览器不同而有所差异，而 POST 方法可以发送任意量的数据。

HTML<form>元素允许通过将<form>元素的 method 属性设置为 GET 或 POST 来指定所需的方法。在提交表单时，<form>元素自动根据 method 属性的规则对<input>元素的数据进行编码。XMLHttpRequest 对象没有这种内置行为，需要开发人员使用 JavaScript 手动创建查询串，其中包含的数据要作为请求的一部分发送给服务器。不论使用的是 GET 请求还是 POST 请求，创建查询串的方法都一样。唯一的区别是当使用 GET 发送请求时必须将查询串追加到请求 url 中，而使用 POST 方法时则在调用 XMLHttpRequest 对象的 send()方法时发送查询串。

下面通过示例演示如何向服务器发送请求参数。首先设计 HTML 页面，在该页面中添加一个输入表单，用于输入姓名、电话和住址。并且再添加两个按钮，单击每个按钮都可以向服务器发送姓名、电话等信息，不过一个使用 GET 方法，另一个使用 POST 方法。当单击按钮时，服务器将输入的信息显示在页面中作为响应。HTML 页面的局部代码如代码 15.1 所示。

代码 15.1　common.html

```
<center>
<form action="#">
```

```
<h2>请详细填写下面信息<h2>
<table>
<tr>
<td>姓名: </td>
<td><input type="text" id="name"/></td>
</tr>
<tr>
<td>电话: </td>
<td><input type="text" id="tele"/></td>
</tr>
<tr>
<td>住址: </td>
<td><input type="text" id="address"></td>
</tr>
</table>
<input type="button" value="使用 GET 方法" onclick="RequestGet()"/>
<input type="button" value="使用 POST 方法" onclick="RequestPOST()"/>
</form>
<br/>
<div id="message" class="div"></div>
</center>
```

在上述代码中，添加一个<div>标记用于显示用户输入的信息。在这里为<div>标记添加 CSS 样式，设置 div 中字体的大小为 12pt，颜色为红色，具体如代码 15.2 所示。

代码 15.2　CSS 样式

```
<style type="text/css">
  .div{
        font-size:12pt;
        color:red;
     }
</style>
```

如果要发送请求，必须创建 XMLHttpRequest 对象。在<head>标记中嵌入 JavaScript 脚本，创建 createXMLHttpRequest()和 createString()函数，分别用于创建 XMLHttpRequest 对象和获取用户输入的信息，具体实现如代码 15.3 所示。

代码 15.3　createXMLHttpRequest()和 createString()函数

```
<script type="text/Javascript">
var xmlHttp;
function createXMLHttpRequest()
{
  //在 IE 下创建 XMLHttpRequest 对象
  try
  {
```

```
        xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
      }
      catch(e)
       {
        try
        {
          xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
      catch(oc)
      {
        xmlHttp = null;
      }
     }
      //在 Mozilla 和 Safari 等非 IE 浏览器下创建 XMLHttpRequest 对象
      if(!xmlHttp && typeof XMLHttpRequest != "undefined")
     {
      xmlHttp = new XMLHttpRequest();
     }
    return xmlHttp;
    }
    //获取用户输入的信息
    function createString()
    {
      var name = document.getElementById("name").value;
      var tele = document.getElementById("tele").value;
      var address = document.getElementById("address").value;
      var str = "name="+name+"&tele="+tele+"&address="+address;
      return str;
    }
    </script>
```

在 createString()函数中，“document.getElementById("name").value”语句表示使用 document 对象的 getElementById()方法获取指定表单元素 name 的值。其他语句依次类推，此处共获取 3 个文本域的值，分别为 name、tele 和 address，即姓名、电话和住址文本框中输入的值。然后将获取的 3 个值组成一个新的字符串 str，每个属性之间用“&”符号隔开，属性和属性值用“=”进行赋值。最后将 str 字符串的值返回。

接下来定义一个 RequestGet()函数用于发送请求，在该函数中使用 GET 方法发送请求，RequestGet()函数如代码 15.4 所示。

代码 15.4　RequestGet()函数

```
    //发送 GET 请求
    function RequestGet()
    {
      //创建 XMLHttpRequest 对象
```

```
    createXMLHttpRequest();
    //定义一个变量获取 PHP 的路径
    var commonString="y.php?";
    commonString = commonString+createString();
    xmlHttp.onreadystatechange=handleStateChange;
    xmlHttp.open("GET",commonString,true);
    xmlHttp.send(null);
}
```

在 RequestGet()函数中，首先调用 createXMLHttpRequest()创建 XMLHttpRequest 对象，并设定服务器程序为 y.php。在 commonString 字符串的末尾添加“?”号，表示后面将会添加参数。“commonString=commonString+createString()”语句表示将 commonString 字符串和 createString()函数的返回值进行连接，组成一个 url。然后调用 onreadystatechange 属性设置处理服务器端响应的函数为 handleStateChange()。最后调用 open()方法指定一个 GET 请求，并指定 url，在这里 url 中包含有编码的参数。send()方法将请求发送给服务器。

除了使用 GET 方法发送请求之外，还可以使用 POST 方法发送请求。POST 与 GET 方法的不同之处在于 POST 允许发送任何格式、任何长度的数据，而 GET 方式只能发送串形式的最大 2KB 数据。接下来再定义一个 RequestPOST()函数，该函数使用 POST 方法发送请求，如代码 15.5 所示。

代码 15.5　RequestPOST()函数

```
//发送 POST 请求
function RequestPOST()
{
   createXMLHttpRequest();
   var url ="y.php";
   var commonString = createString();
   xmlHttp.open("POST",url,true);
   xmlHttp.onreadystatechange=handleStateChange;
   xmlHttp.setRequestHeader("Content-Type","application/x-www-form-
   urlencoded;");
   xmlHttp.send(commonString);
}
```

在 RequestPOST()函数中，同样首先调用 createXMLHttpRequest()函数，并设定 url 的值为 y.php。然后使用 open()方法发送一个请求，这一次指定的请求方法是 POST，另外指定没有追加“名称/值对”的 url。并设定 onreadystatechange 的值为 handleStateChange()函数，所有响应会以 GET 方法相同的方式得到处理。为了确保服务器中知道请求体中有请求参数，需要调用 setRequestHeader()，将 Content-Type 值设置为“application/x-www-form-urlencode;”。最后调用 send()方法，并将查询串作为参数传递给这个方法。

无论使用 GET 方法，还是使用 POST 方法传递，其处理服务器端响应信息的函数相同，处理请求的 handleStateChange()函数如代码 15.6 所示。

代码 15.6　handleStateChange()函数

```
//回调函数
function handleStateChange()
  {
     if (xmlHttp.readyState == 4)
     {
     //判断对象状态
        if (xmlHttp.status == 200)
        {
          message.innerHTML = xmlHttp.responseText;
        }
     }
  }
```

接下来编写 PHP 服务器页面，用于实现服务器端程序，实现处理信息并返回，具体实现如代码 15.7 所示。

代码 15.7　y.php

```
<?php
$name = $_REQUEST["name"];
$tele = $_REQUEST["tele"];
$address = $_REQUEST['address'];
echo("你好，你填写的信息如下所示：");
echo("<br/>");
echo("姓名是："．$name);
echo(",电话是："．$tele);
echo(",住址是："．$address);
?>
```

至此程序已经设置完成，接下来在浏览器中查看运行效果。在浏览器地址栏中输入"http:localhost:8080/common.html"之后，则会显示相应的页面。在文本框中输入信息，单击【使用 GET 方法】或者【使用 POST 方法】按钮，运行结果如图 15-1 所示。

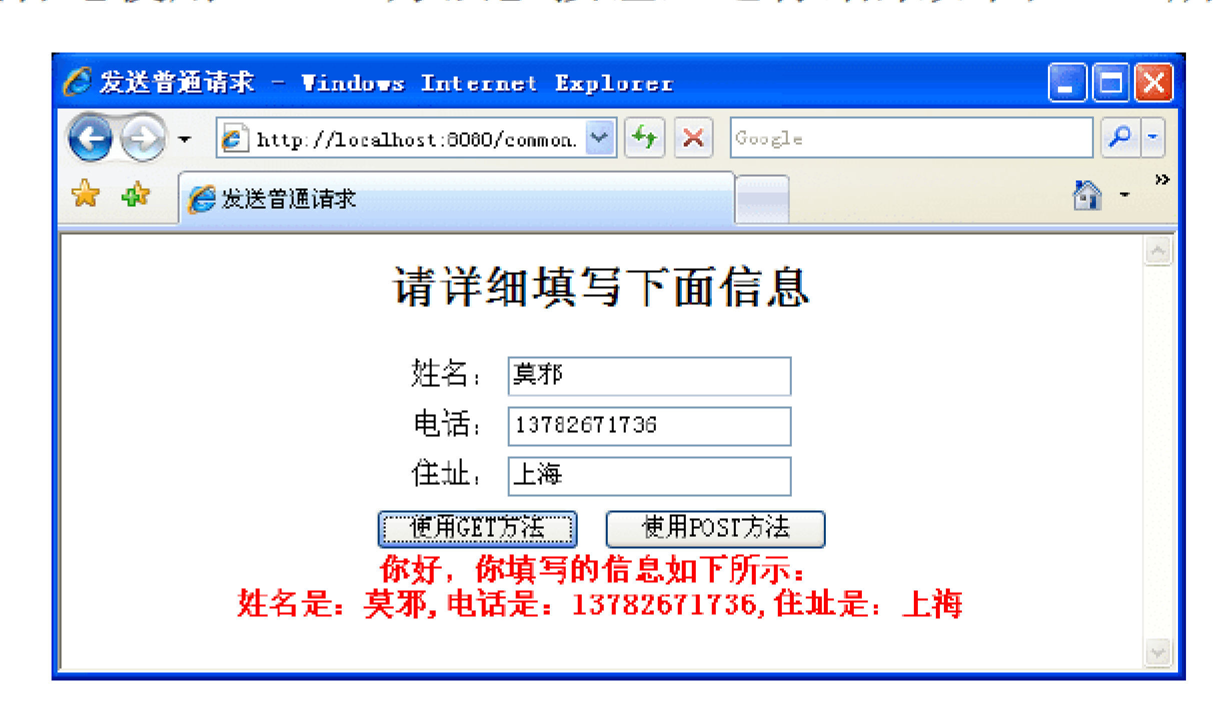


图 15-1　发送普通请求



绝大多数 HTTP 请求中，转义序列%20 用于表示一个空格，文本“Anne is a girl”将以 Anne%20is%20a%20girl 的形式通过 HTTP 发送。

15.1.2　请求参数作为 XML 发送

在上面示例中讲解了使用普通文本发送请求，接下来讲解如何向服务器发送 XML 文件。由于 XML 本身的优势及浏览器支持的特性，可以将 XML 作为请求体的一部分发送到服务器，然后服务器可以从请求体中读取到 XML 并进行处理，返回相应的结果。

下面通过示例讲解如何使用 Ajax 向服务器发送 XML 文件，并且通过 PHP 服务器页面执行对 XML 文件的查询，然后将查询的结果显示在 HTML 页面中。在该示例中需要一个 HTML 页面、一个 XML 文件和一个 PHP 页面。在 XML 文件中包含小说的信息，在 HTML 页面中单击按钮之后，则会在页面中显示该小说的详细信息。首先编写 XML 文件，如代码 15.8 所示。

代码 15.8　novel.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<novels>
  <novel>
    <name>五星饭店</name>
    <author>海岩</author>
    <price>28.00 </price>
    <introduce>
    这是一本关于一群年轻人奋斗的过程，在追求属于自己的真实人生
    </introduce>
    <publish>现代出版社</publish>
    <publishDate>2006-08-01 </publishDate>
  </novel>
  <novel>
    <name>你的生命如此多情</name>
    <author>海岩</author>
    <price>23.00</price>
    <introduce>
    林星的男友刘文庆不惜用女友的美貌来接近长天集团的上层人物，愤怒的林星找到了一个报复的
    办法……
    </introduce>
    <publish>作家出版社</publish>
    <publishDate>2000-10-01</publishDate>
  </novel>
</novels>
```

HTML 页面中只需要添加一个 button 按钮和<div>标记。当单击按钮时触发 onclick 事件，则调用发送请求的函数。请求完成之后，则使用回调函数处理请求。HTML 页面如代码 15.9

所示。

代码 15.9　novel.html

```
<html>
<head>
<meta xmlns-equiv="Content-Type" content="text/html; charset=UTF-8">
<script type="text/javascript">
var xmlHttp;
//创建 XMLHttpRequest 对象
//createXMLHttpRequest()函数在这里不再给出代码，可以参照代码 15.3
//发送请求
function showBook()
{
  createXMLHttpRequest();
  var url="novel.php";
  xmlHttp.onreadystatechange=StateChange;
  xmlHttp.open("GET",url,true);
  xmlHttp.send(null);
}
//处理请求
function StateChange()
{
  if(xmlHttp.readyState==4)
  {
     if(xmlHttp.status==200)
    {
     document.getElementById("message").innerHTML= xmlHttp.responseText;
    }
  }
}
</script>
</head>
<body>
<input type="button" name="btn" id="button1" value="单击按钮显示详细信息"
onclick="showBook()">
<div id="message"></div>
</body>
</html>
```

接下来开始编写 PHP 服务器页面。在该页面中首先创建 XML 文件的 XML DOM 对象，然后使用 load()方法加载 XML 文件到内存中，并且将 XML 文件中的所有信息都读取出来，PHP 页面如代码 15.10 所示。

代码 15.10　novel.php

```
<?php
```

```
    $doc = new DOMDocument();
    $doc->load('novel.xml' );
    $novels = $doc->getElementsByTagName( "novel" );
    foreach( $novels as $book )
    {
      $names = $book->getElementsByTagName("name");
      $name = $names->item(0)->nodeValue;
      $authors = $book->getElementsByTagName("author");
      $author = $authors->item(0)->nodeValue;
      $prices = $book->getElementsByTagName("price");
      $price = $prices->item(0)->nodeValue;
      $introduces = $book->getElementsByTagName("introduce");
      $introduce = $introduces->item(0)->nodeValue;
      $publishs = $book->getElementsByTagName("publish");
      $publish = $publishs->item(0)->nodeValue;
      $publishDates = $book->getElementsByTagName("publishDate");
      $publishDate = $publishDates->item(0)->nodeValue;
      echo ("书名: ".$name."<br/>");
      echo ("作者: ".$author."<br/>");
      echo ("价格: ".$price."<br/>");
      echo ("简介: ".$introduce."<br/>");
      echo ("出版社: ".$publish."<br/>");
      echo ("出版日期: ".$publishDate."<br/>");
      echo ("<hr/>");
    }
  ?>
```

此时程序已经设置完成，接下来在浏览器中查看运行效果。在浏览器地址栏中输入“http://localhost:8080/novel.html”之后，页面中显示一个按钮。当单击该按钮时则会将 XML 文件中的数据读取到 HTML 页面中，页面运行效果如图 15-2 所示。

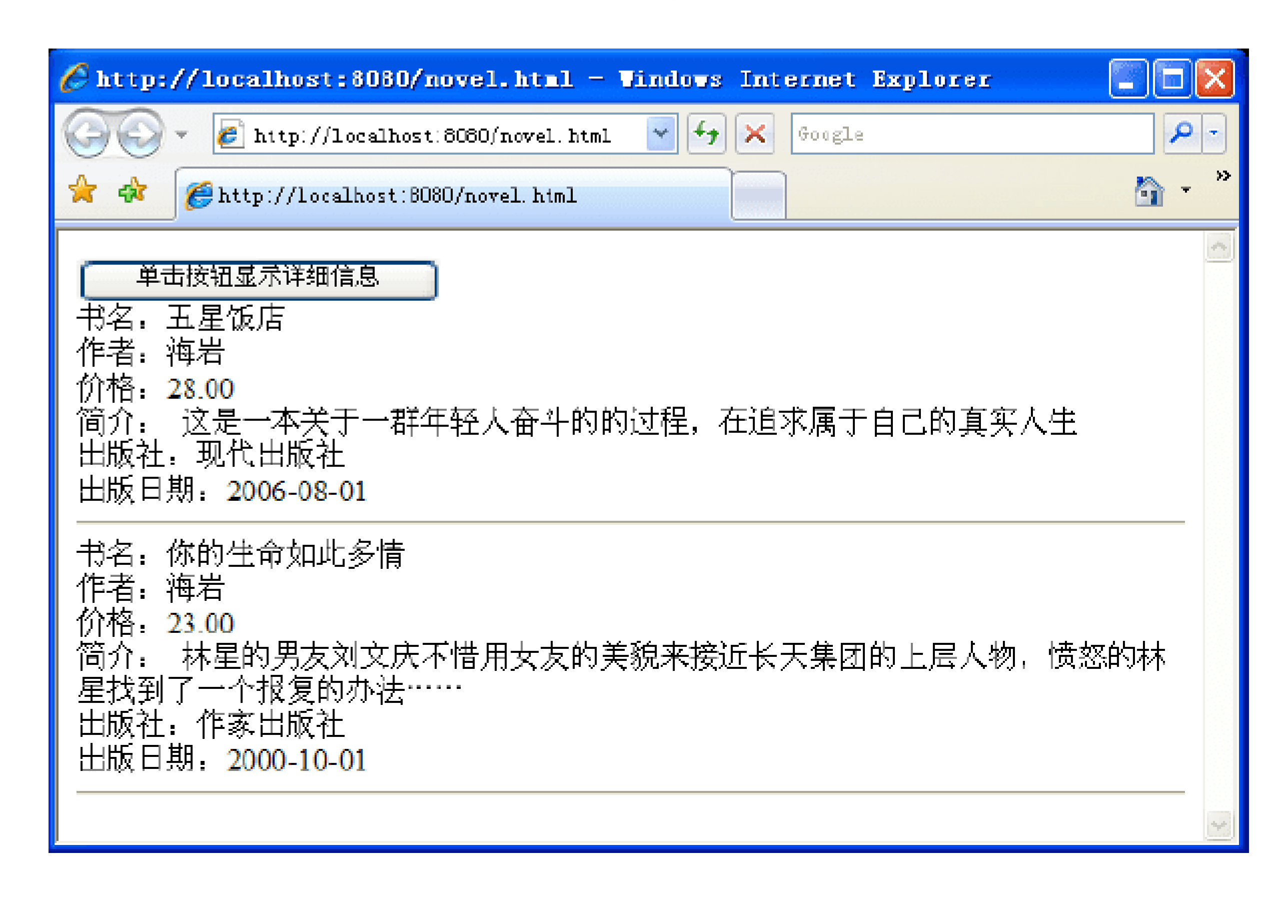


图 15-2　响应 XML 请求

15.1.3 发送 JSON 格式请求

JSON（JavaScript Object Notation）是一种轻量级的数据交换格式，比 XML 更轻巧。JSON 是 JavaScript 的原生格式，这意味着在 JavaScript 中处理 JSON 数据不需要任何特殊的 API 或工具包。

JSON 的使用规则很简单：对象是一个无序的“名称/值对”集合。一个对象以“{”开始，以“}”结束。每个“名称”后跟一个冒号“:”;“名称/值对”之间使用逗号“,”分隔。

下面通过 XML 文件比较 JSON 格式的使用。例如在 XML 文件中包括姓名、年龄、密码和个人主页。XML 文件的内容如代码 15.11 所示。

代码 15.11 uers.xml

```
<?xml version="1.0" encoding="GB2312"?>
<user>
<name>Baby</name>
<age>21</age>
<password>123456</password>
<website>http://www.itzcn.com</website>
</user>
```

将 XML 文件中的内容以 JSON 格式显示，如代码 15.12 所示。

代码 15.12 JSON 格式显示 XML 中的数据

```
var user =
{
  "name":"Baby",
  "age":"21",
  "password":"123456",
  "website":" http://www.itzcn.com ",
}
```

与 XML 一样，JSON 也是基于文本的，而且都使用 Unicode 编码，同样具有可读性。XML 比较适合于标记文档，而 JSON 却更适合于进行数据交换处理。下面通过示例说明如何发送 JSON 格式的参数请求。首先需要创建一个 XMLHttpRequest 对象，然后再编写一个发送请求的函数和一个处理请求的函数，页面的具体实现如代码 15.13 所示。

代码 15.13 myjson.html

```
<html>
<head>
<title>发送 JSON 请求</title>
<script type="text/javascript">
var xmlHttp;
//创建 XMLHttpRequest 对象
```

```
//createXMLHttpRequest()函数在这里不再给出代码，可以参照代码 15.3
//发送 JSON 格式请求
function sendRequestJosn()
{
  var url="myjson.php";
  createXMLHttpRequest();
  xmlHttp.open("GET",url,true);
  xmlHttp.onreadystatechange=getJSON;
  xmlHttp.send(null);
}
//getJSON()函数解析 JSON 字符串
function getJSON()
{
  if (xmlHttp.readyState == 4) {
     if (xmlHttp.status == 200) {
     var arr = xmlHttp.responseText;
     var json = eval("(" + arr + ")");
     var result="";
     result+="姓名: "+json.name+"<br/>";
     result+="年龄: "+json.age+"<br/>";
     result+="密码: "+json.password+"<br/>";
     result+="邮箱: "+json.cation.email+"<br/>";
     result+="主页: "+json.cation.website;
     document.getElementById('message').innerHTML = result;
     }
   }
}
</script>
</head>
<body onload="sendRequestJosn()">
<h3>用户信息如下所示: </h3>
<div id="message"></div>
</body>
</html>
```

在 getJSON()函数中声明一个 arr 变量用于保存从服务器获取的所有数据，然后使用 eval()方法将获取的数据生成执行语句，即将变量 json 作为对象名称。接下来再定义一个变量，使用 json 对象将字符串中的数据一条一条的读取出来，然后将信息显示在 id 为 message 的<div>标记中。在页面的 body 中触发 onload 事件，表示打开该页面则会向服务器发送请求。

客户端页面设置好以后，接下来开始设置服务器端页面。在服务器页面中声明一个 JSON 字符串，并且为该字符串赋值。接着使用 json_encode 进行对 JSON 字符串编码，最后输出 JSON 字符串。服务器页面的具体实现如代码 15.14 所示。

代码 15.14　myjson.php

```
<?php
header('Content-type:text/html;charset=GBK');
```

```
//生成 JSON 字符串
$arr = array(
    'name'=>'Baby',
    'age'=>'21',
    'password'=>'123456',
    'cation' => array(
    'email' => 'loveBaby1314@163.com',
    'website' =>'http://www.itzcn.com',
    )
   );
//使用 json_encode 进行编码
$message = json_encode($arr);
//JSON 字符串
echo $message;
?>
```

至此整个程序已经设置完成，接下来在浏览器中查看页面运行效果。在浏览器地址栏中输入“http://localhost:8080/myjson.html”之后，则触发 onload 事件。页面运行效果如图 15-3 所示。



图 15-3　处理 JSON 参数

15.2　处理服务器响应

在发送请求之前通过一个回调函数指明如何处理服务器的响应，即将 XMLHttpRequest 对象的 onreadystatechange 属性设置为将要运行的函数名。这样当服务器处理完请求之后就会自动调用该函数，并且不需要担心该函数的任何参数，不过需要考虑 HTTP 的就绪状态和返回的状态码。

在 Ajax 应用程序中，HTTP 的就绪状态表示请求的状态和情形，用于确定请求是否已经开始、是否得到响应或者“请求/响应”模式是否已经完成，还可以帮助确定读取服务器提供的响应文本或者数据是否安全。HTTP 的就绪状态有 5 种，分别是 0、1、2、3、4，在前面章节中已经介绍了它们的含义，在这里不再多述。

除了就绪状态外，还需要检查 HTTP 的状态，指明请求是否已经成功，并且还可以揭示

请求失败的原因。当 HTTP 的状态为 200 时，表示服务器已经成功的接受客户端的请求。

在 Ajax 客户端向服务器发送请求后，服务器端根据具体的请求返回对应的数据。XMLHttpRequest 对象提供两个用于处理服务器返回数据的属性：第一个属性 responseText 将响应转换成字符串，第二个属性 responseXML 将响应转换成 XML 文档对象。

15.2.1　处理文本格式的响应

对于大多数的普通文本请求，可以使用 ResponseText 来接收响应内容。这样一来便可以很容易使用 JavaScript 调用 HTML 元素的 innerHTML 方法向页面中插入一段文字或标记，如果使用 innerText 方法则可以插入一段文本。

下面通过示例讲解如何处理普通的文本格式响应。首先需要创建一个 XMLHttpRequest 对象，然后编写发送请求的函数，在该函数中指定处理请求的函数，具体实现如代码 15.15 所示。

代码 15.15　text.html

```
<html>
<head>
<title>普通文本响应</title>
<script type="text/javascript">
var xmlHttp;
//createXMLHttpRequest()函数在这里不再给出代码，可以参照代码 15.3
//发送请求
function sendRequest()
{
  var url="text.php";
  createXMLHttpRequest();
  xmlHttp.onreadystatechange=handleStateChange;
  xmlHttp.open("GET",url,true);
  xmlHttp.send(null);
}
//处理请求
function handleStateChange()
{
  if(xmlHttp.readyState==4)
  {
     if(xmlHttp.status==200)
    {
      message.innerHTML = xmlHttp.responseText;
    }
  }
}
</script>
</head>
```

```
<body>
<input type="button" id="sumbit" name="button1" value="单击显示数据" onclick=
"sendRequest()">
<div id="message"></div>
</body>
</html>
```

在 sendRequest()函数中声明一个变量，用于保存 PHP 服务器页面的路径。然后创建 XMLHttpRequest 对象，并指定回调函数，最后发送请求。在 handleStateChange()函数中，首先判断 HTTP 的状态，如果响应完成则使用 responseText 属性将响应转换为字符串，并且在<div>标签中显示。

在 PHP 服务器页面中定义 3 个变量，分别用于存储新闻的标题、发表时间和内容。然后再使用输出语句将其输出，具体实现如代码 15.16 所示。

代码 15.16　text.php

```
<?php
header('Content-type:text/html;charset=GBK');
$title="房价反弹，房市回暖";
$date="2008-11-25";
$content="四万亿入市决定后，政府最愿意看到的是什么？新近笔者赴北京边的一个城市搞调查，与当地国土资源局与房地局建委等的访谈中，他们对于人们对房价的过于专注感到惊讶，他们说“人们的消费心理是买涨不买跌，看跌不买跌”，无论房价如何下行，消费不启动，一切都会白费。";
echo("标题：".$title);
echo("<br/>");
echo("发表时间：".$date);
echo("<br/>");
echo ("发表内容：".$content);
?>
```

此时程序已经设置完成，接下来在浏览器中查看运行效果。必须保证 PHP 服务器是打开的，然后在浏览器地址栏中输入“http://localhost:8080/text.html”之后，则可以在页面中看到一个按钮，单击该按钮则会将 PHP 页面的数据显示在该页面中，运行效果如图 15-4 所示。

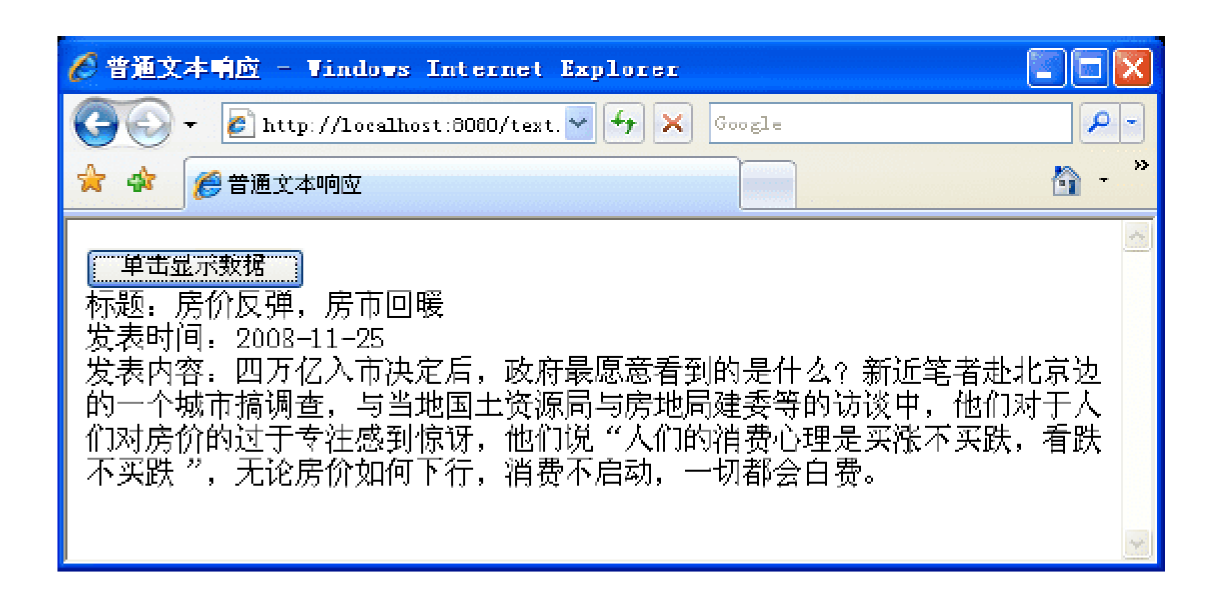


图 15-4　普通文本响应

在使用 XMLHttpRequest 对象的 ResponseText 方法获取服务器的响应结果时，返回的是普

通文本。此时服务器无须对返回的结果数据进行必要编码，除非开发人员自定义了相应的格式。

使用 ResponseText 方法，返回的结果视为普通文本，其往往被当作一个字符串，然后再对其进行如下之一的处理。

- 将这个字符串作为 HTML 标记的内容，然后使用 innerHTML 方法直接添加到控件里在页面上显示。
- 将这个字符串作为用户自定义的某种格式，再使用 JavaScript 函数对其进行转换。
- 这个字符串直接被 JavaScript 解析后加入相应的 HTML 代码中。

总之，要么将返回的普通文本作为一个整体显示在页面的适当位置，要么 JavaScript 对返回的普通文本做进一步的解析处理或者编码后再呈现在页面上。所以如下两种情况经常会使用 ResponseText 获取服务器的响应结果数据。

- 返回的响应数据相对比较简单，不需要经过解析处理，直接作为一个字符串显示在页面上。这时可以使用 ResponseText，获取后使用 innerHTML 赋值等方式显示，常见的这种情况有数据校验结果、系统界面帮助、系统操作提醒以及列表正文内容等。
- 返回的响应数据结构比较简单，使用特殊的分隔符号或者简单的 JavaScript 脚本就可以分隔开，JavaScript 可以快速对其进行处理。

15.2.2　处理 XML 格式的响应

在服务器端，对于复杂的数据结构数据通常使用 XML 文件格式返回。XML 文件中的数据就是操作的重点，这些 XML 数据可以预先设定，也可以来自于数据库表或文件。对于这些不能预先设定好的数据，通常需要在服务器端临时自动生成 XML 文件。

下面通过示例说明如何在服务器端创建 XML 文件，并在客户端显示 XML 文件中的数据。首先需要定义 HTML 页面，在该页面中创建 XMLHttpRequest 对象、发送请求和处理请求，具体实现如代码 15.17 所示。

代码 15.17　employee.html

```
<html>
<head>
<title>发送 XML 请求</title>
<script type="text/javascript">
var xmlHttp;
//创建 XMLHttpRequest 对象
//createXMLHttpRequest()函数在这里不再给出代码，可以参照代码 15.3
//发送请求
function startRequest()
{
   createXMLHttpRequest();
   xmlHttp.onreadystatechange=handleStateChange;
   xmlHttp.open("GET","employee.php",true);
   xmlHttp.send(null);
```

```
    }
    function handleStateChange()
    {
       if(xmlHttp.readyState==4)
      {
         if(xmlHttp.status==200)
        {
           var employee = xmlHttp.responseXML;
           var emp = employee.getElementsByTagName("employee ");
           var str = "<table border='1' width='45%'><tr><td>姓名</td><td>年龄
           </td><td>住址</td></tr>";
           for(var i=0;i< emp.length;i++)
           {
             str+="<tr><td>"+ emp[i].childNodes[0].firstChild.data+"</td>";
             str+="<td>"+ emp[i].childNodes[1].firstChild.data+"</td>";
             str+="<td>"+ emp[i].childNodes[2].firstChild.data+"</td>";
             str+="</tr>";
           }
           str+="</table>";
           document.getElementById("message").innerHTML=str;
        }
      }
    }
    </script>
    </head>
    <body onload="startRequest()">
    <form action="#">
    <center>
    <h3>职工信息</h3>
    <div id="message"></div>
    </center>
    </body>
    </html>
```

在 handleStateChange()函数中获取 XML 文件中的根节点，然后再创建一个表格，使用 for 循环遍历节点中的元素，并且将遍历的元素读取到表格中。最后将表格中的数据显示在 id 为 message 的<div>标签中。客户端设计好以后，接下来设计服务器端页面。在服务器端生成 XML 文件，具体实现如代码 15.18 所示。

代码 15.18　employee.php

```
    <?php
    header('Content-Type: text/xml');
    $xml="<?xml version='1.0' encoding='GB2312'?>";
    $data="<employees>
    <employee><name>江南</name><age>22</age><address>江西</address></employee>
```

```
<employee><name>简凝</name><age>22</age><address>北京</address></employee>
</employees>";
echo $xml.$data;
?>
```

客户端和服务器端都设置好以后，接下来就可以在浏览器中查看运行效果。在浏览器地址栏中输入“http:localhost:8080/employee.html”之后，页面运行效果如图 15-5 所示。

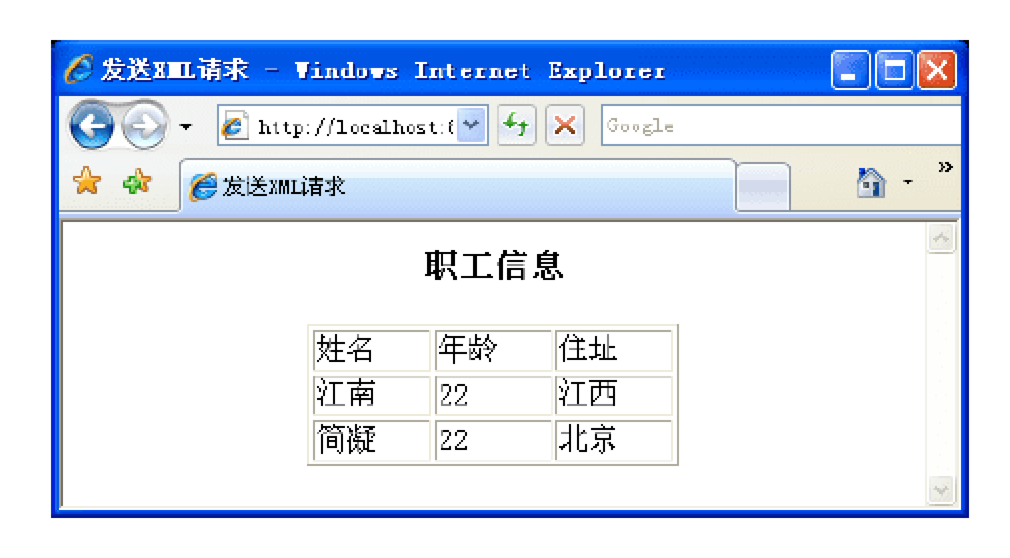


图 15-5　显示 XML 文件



如果页面在运行时出现乱码情况，需要将文件“另存为”时，在编码下拉列表框中选择 UTF-8 即可。

15.2.3　处理 JSON 格式的响应

下面通过示例讲解在客户端页面传递参数到服务器页面，并在服务器端生成 JSON 格式的字符串，然后将生成的 JSON 字符串再次返回到 HTML 页面。在该示例中当用户单击超级链接查看用户主页时，则会弹出一个对话框。在对话框中根据传递的参数生成个人主页的网址，HTML 页面如代码 15.19 所示。

代码 15.19　mywebsite.html

```
<html>
<head>
<script type="text/javascript">
var xmlHttp;
//创建 XMLHttpRequest 对象
//createXMLHttpRequest()函数在这里不再给出代码，可以参照代码 15.3
//发送请求
function sendRequest(method,url,id)
{
  createXMLHttpRequest();
  var url1 = "mywebsite.php?url="+url+"&id="+id+"&rand="+new Date().
  getTime();
  xmlHttp.open(method,url1,true);
  xmlHttp.onreadystatechange=getJSON;
  xmlHttp.send(null);
```

```
}
function getJSON()
{
  if (xmlHttp.readyState == 4) {
     if (xmlHttp.status == 200) {
       var json=eval('('+xmlHttp.responseText+')');
       alert(json.id+"的主页是: "+json.url);
     }
   }
}
</script>
</head>
<body>
<a href="#" onclick="sendRequest('get','myDBaby.com','Baby')">单击此处，查看
Baby 的主页</a>
</body>
</html>
```

在上述代码中，sendRequest()函数定义 3 个参数，用于向服务器页面发送数据信息。然后调用 createXMLHttpRequest()函数创建 XMLHttpRequest 对象，接着定义服务器的 url 并且为其传递参数。通过给定的 url 打开 XMLHttpRequest 对象，最后发送 HTTP 请求。在 getJSON()函数中使用 eval()函数将从服务器获取的信息转换为 json 对象，然后以对话框的形式显示信息。

HTML 页面设置好以后，接下来需要设置 PHP 服务器页面。在该页面中定义两个变量用于获取从客户端页面传递的参数，接着定义一个 JSON 格式的字符串，并且将传递过来的参数赋值给 JSON 字符串，最后输出 JSON 字符串。PHP 页面如代码 15.20 所示。

代码 15.20 mywebsite.php

```
<?php
$url=$_GET['url'];
$id=$_GET['id'];
$arr=array(
'id'=>$id,
'url'=>'http://www.'.$url,
);
$mywebsite=json_encode($arr);
echo $mywebsite;
?>
```

此时程序已经编写完成，接下来在浏览器中查看运行效果。在浏览器地址栏中输入“http://localhost:8080/mywebsite.html”之后，单击超级链接则会弹出一个对话框，页面运行效果如图 15-6 所示。

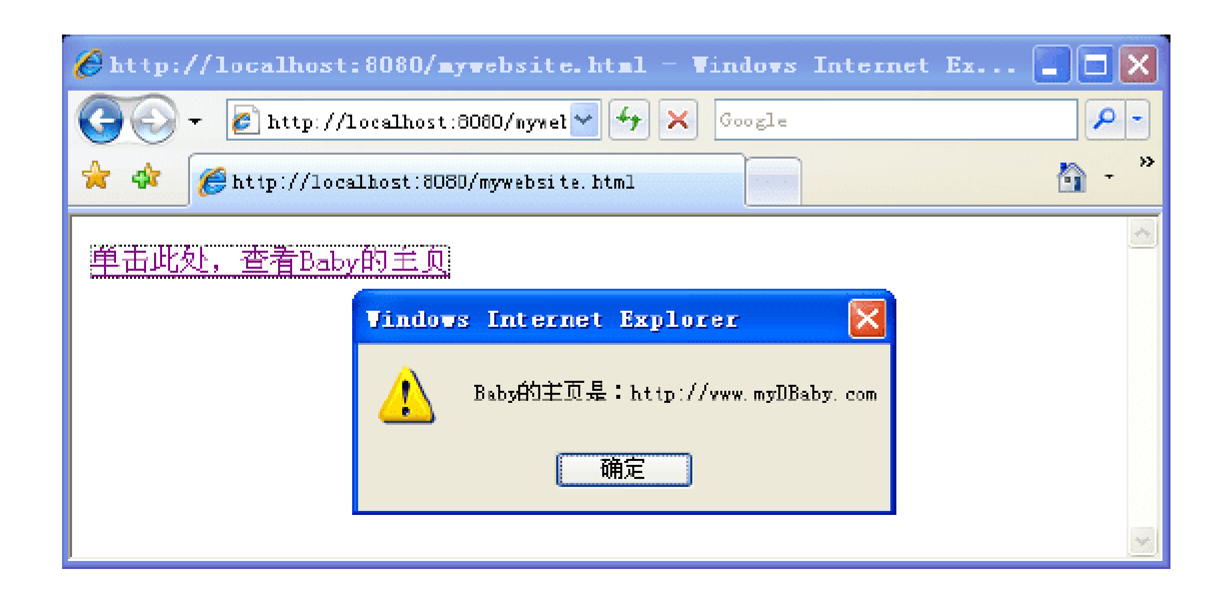


图 15-6　处理 JSON 参数

15.3　Ajax 实例

下面通过使用 Ajax 和 PHP 控制图片的显示。在 HTML 页面中添加一个 CSS 样式表，设置在页面的主题中字体的大小为 12px，并设置超级链接字体的颜色和单击后显示的颜色。然后在 JavaScript 脚本语言中编写发送 XMLHttpRequest 请求和处理请求的函数。HTML 页面如代码 15.21 所示。

代码 15.21　showPic.html

```
<html>
<head>
<title>无刷新显示图片</title>
<style type="text/css">
<!--
body {
 font-size: 12px;
}
a:link, a:visited, a:active {
 color: #616161;
 text-decoration: none;}
a:hover {
 text-decoration: none;
 background-color: #616161;
 color: #FFFFFF;}
-->
</style>
<script language="javascript">
<!--
function InitAjax(){
if (window.ActiveXObject) {
xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

```
else if (window.XMLHttpRequest) {
xmlHttp = new XMLHttpRequest();
}
return xmlHttp;
}
function getPic(pic_id){
  if(typeof(pic_id)=='undefined'){
   return false;
  }
 var url="showPic.php?pic_id="+pic_id;
 var show=document.getElementById("show_Layer");
 var ajax=InitAjax();
 ajax.open("GET",url,true);
 ajax.onreadystatechange=function(){
   if(ajax.readyState==4 && ajax.status==200){
    show.innerHTML=ajax.responseText;
   }
 }
 ajax.send(null);
}
-->
</script>
</head>
<body>
<a href="#" onclick="getPic(1)">图片-1</a>  
<a href="#" onclick="getPic(2)">图片-2</a>  
<a href="#" onclick="getPic(3)">图片-3</a>  
<a href="#" onclick="getPic(4)">图片-4</a>  
<br>
<div id="show_Layer"></div>
</body>
</html>
```

在 JavaScript 脚本语言中定义两个函数，InitAjax()用于创建 XMLHtttpRequest 对象，getPic()函数用于发送 XMLHttpRequest 请求。在 getPic()函数中定义一个参数，用于传递图片的名称。然后定义一个 url 变量，用于保存服务器页面的路径，并为其传递一个参数。接着创建 XMLHttpRequest 对象，并通过给定的 url 打开 XMLHttpRequest 对象。使用 onreadystatechange 属性指定一个函数，用于处理完成的请求，最后发送请求。

在 PHP 页面中获取图片的名称，然后定义图片输出时的大小，PHP 页面如代码 15.22 所示。

代码 15.22　showPic.php

```
<?php
echo "<img src='pic_".$_GET[pic_id].".jpg' width="200" border="0">";
```

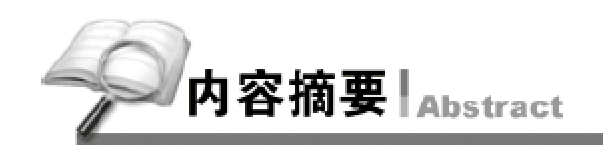
```
?>
```

此时程序已经设置完成，接下来在浏览器中查看页面运行效果。在浏览器地址栏中输入“http://localhost:8080/showPic.html”之后，可以在页面中看到一些超级链接。单击【图片-1】超级链接，页面运行效果如图 15-7 所示。



图 15-7　显示图片

第16章 Ajax 设计模式



内容摘要 | Abstract

设计模式使人们可以更加简单、方便地重复使用成功的设计和体系结构，是一套被反复使用、经过良好分类的设计经验的总结。设计模式能够提高代码的重用率，令代码更容易被理解，保证代码的可靠性。而在 Ajax 中，也需要利用设计模式来更加快速、有效地解决开发中的一些问题。本章将介绍如何在 Ajax 应用中有效地使用设计模式，并详细介绍 MVC 模式在 Ajax 中的具体应用。

学习目标 | Objective

- 理解什么是设计模式
- 了解设计模式的构成和原则
- 了解基本的 GOF 设计模式
- 了解 Ajax 中常用的设计模式
- 了解如何按设计模式重构 Ajax
- 掌握在 Ajax 中如何应用 MVC 设计模式

16.1 设计模式

模式的概念诞生于 20 世纪 70 年代，当时是用来描述建筑规划问题的解决方案，近年来软件开发领域借用了这一概念。目前在服务器端的 Java 开发有很多优秀的设计模式，如 Struts，微软也在.NET Framework 中大力推动设计模式的应用。从根本上讲，设计模式仅仅是用来描述在软件设计中解决特定问题的一种可重复使用的解决方案。

16.1.1 设计模式概述

设计模式无处不在。在阅读技术方面的出版物或者浏览技术方面的网站时，很容易发现对设计模式的引用。模式不是框架（Framework），也不是过程。模式不是简单的“问题的解决方案”，因为模式必须是典型问题的解决方案。模式不能套用，不要以为在任何一个系统中都要使用某些设计模式。系统的设计也不是含有设计模式就好，更不是含有越多的设计模式就越好。

设计模式提供了一种共享经验的方式，可以使团体受益和避免不断的重复发明。设计模式通常捕捉问题的描述、问题的语境、推荐的问题解决方案以及使用解决方案后可以预见到的结果。为了具有最广泛的适用性（从而对更多的读者有用），设计模式通常从取决于环境的精确细节中抽象出来。这种抽象性产生了一些将设计模式应用到现有的案例中所必需的译码。这是一个重要细节：尽管设计模式是共享专业知识的好方法，但通常其对正确应用专业知识是非常重要的。

现在可复用面向对象软件系统一般划分为 3 大类：应用程序、工具箱和框架。平时开发的具体软件都是应用程序，Java 的 API 属于工具箱，而框架是构成一类特定软件可复用设计的一组相互协作的类，EJB（Enterprise Java Beans）是 Java 应用于企业计算的框架。

框架通常定义了应用体系的整体结构以及类和对象的关系等设计参数，以便于具体应用实现者能集中精力于应用本身的特定细节。框架主要记录软件应用中共同的设计决策，框架强调设计复用，因此框架设计中必然要使用设计模式.

另外设计模式有助于对框架结构的理解，成熟的框架通常使用了多种设计模式。如果读者熟悉这些设计模式，毫无疑问将可以迅速掌握框架的结构。一般的开发者如果第一次接触 EJB 等框架，会觉得特别难学，不容易掌握，那么此时如果先掌握设计模式，无疑为学习框架打下结实的基础。

16.1.2　设计模式组成要素和原则

通常定义一个模式需要具备如下 4 个基本要素。

- **模式名称**

一个助记名，用一两个词来描述模式的问题、解决方案和效果。命名一个新的模式增加设计词汇。设计模式允许在较高的抽象层次上进行设计。基于一个模式词汇表，读者自己以及同事之间就可以讨论模式并在编写文档时使用它们。模式名可以帮助思考，便于与其他人交流设计思想以及设计结果。找到恰当的模式名也是设计模式编目工作的难点之一。

- **问题**

描述了应该在何时使用模式。解释了设计问题和问题存在的前因后果，可能描述了特定的设计问题，例如怎样用对象表示算法等，也可能描述了导致不灵活设计的类或对象结构。有时问题部分会包括使用模式必须满足的一系列先决条件。

- **解决方案**

描述了设计的组成成分，它们之间的相互关系及各自的职责和协作方式。因为模式就像一个模板，可应用于多种不同场合，所以解决方案并不描述一个特定而具体的设计或实现，而是提供设计问题的抽象描述和怎样用一个具有一般意义的元素组合（类或对象组合）来解决这个问题。

- **效果**

描述了模式应用的效果及使用模式应权衡的问题。尽管描述设计决策时，并不总提到模式效果，但它们对于评价设计选择和理解使用模式的代价及好处具有重要意义。软件效果大多关注对时间和空间的衡量，它们也表述了语言和实现问题。因为复用是面向对象设计的要素之一，所以模式效果包括它对系统的灵活性、扩充性或者可移植性的影响，显式地列出这

些效果对理解和评价这些模式很有帮助。

现在越来越多的开发人员开始注意设计模式，了解设计模式并使用它们。与此同时，产生的问题也越来越多，像为什么要用设计模式、设计模式为什么要这么设计、为什么要提倡“设计模式”等。这些问题的根本原因之一是为了代码复用，增加代码可维护性。那么如何才能实现代码复用？

首先来了解一些设计模式的原则：“开一闭”原则（Open Closed Principal）、里氏代换原则和合成复用原则等。设计模式就是实现了这些原则，从而达到了代码复用、增加可维护性的目的。这些原则的简要介绍如下所示。

（1）“开一闭”原则

此原则是由 Bertrand Meyer 提出的。就是说模块应对扩展开放，而对修改关闭。模块应尽量在不修改原（是“原”而不是“源”，指原来的代码）代码的情况下进行扩展。这是代码重用的最基本原则，也是核心原则。

该原则带来了两个无可比拟的优越性。

- 通过扩展已有的软件系统，可以提供新的行为以满足对软件的新需求，使变化中的软件系统有一定的适应性和灵活性。
- 已有的软件模块，特别是最重要的抽象层模块不能再修改，这就使变化中的软件系统有一定的稳定性和延续性。

（2）里氏代换原则

里氏代换原则提出子类可以代替父类出现在父类能出现的地方。Java 中的上转型对象就是里氏代换原则的一个体现，可以说里氏代换原则是继承复用的一个基础。

（3）合成复用原则

尽量少用继承，多用合成关系来实现。也就是类之间尽量是通过消息机制建立联系，而不是通过继承。

（4）依赖倒转原则

该原则的目的是降低类之间的耦合度。最常见的表述方式有两种：一种表述是“抽象不应该依赖于细节，细节应当依赖于抽象”；另一种表述是“要针对接口编程，而不是针对实现编程”。

（5）接口隔离原则

使用多个专门的接口比使用单一的总接口要好。也就是说，一个类对另外一个类的依赖性应当是建立在最小的接口上的。

（6）抽象类

抽象类不能用于创建实例、为子类提供公用的属性和方法以及规定子类所必须实现的方法。

总而言之，一个对象应当对其他对象有尽可能少的了解。如果两个类不必彼此直接通信，那么这两个类就不应当发生直接的相互作用。如果其中的一个类需要调用另一个类的某一个方法，可以通过第三者转发这个调用。设计模式就是对这些原则的一个实现，下一小节将介绍符合这些原则的基本设计模式。

16.1.3　基本设计模式

程序设计是思维具体化的一种方式，是思考如何解决问题的过程。设计模式是在解决问题的过程中一些良好思路的经验集成。一提到设计模式，人们总会提到 GOF 的著作，因为他最先列出了 23 种模式并集合在一起进行说明。

GOF 的“设计模式”同时也是第一次将设计模式提升到理论高度，并将之规范化。下面将简要介绍 GOF 所提到的这些设计模式。

- **Abstract Factory 模式**　提供一个创建一系列相关或相互依赖对象的接口，而无须指定其具体的类。
- **Adapter 模式**　将一个类的接口转换成客户希望的另外一个接口。Adapter 模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。
- **Bridge 模式**　将抽象部分与它的实现部分分离，使它们都可以独立地变化。
- **Builder 模式**　将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。
- **Chain of Responsibility 模式**　为解除请求的发送者和接收者之间耦合，而使多个对象都有机会处理这个请求。将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象将其处理。
- **Command 模式**　将一个请求封装为一个对象，从而可用不同的请求对客户进行参数化。对请求排队或记录请求日志，以及支持可取消的操作。
- **Composite 模式**　将对象组合成树形结构以表示“部分-整体”的层次结构。这种模式使得客户对单个对象和复合对象的使用具有一致性。
- **Decorator 模式**　动态地给一个对象添加一些额外的职责。就扩展功能而言，这种模式比生成子类方式更为灵活。
- **Facade 模式**　为子系统中的一组接口提供一个一致的界面。Facade 模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。
- **Factory Method 模式**　定义一个用于创建对象的接口，让子类决定将哪一个类实例化。Factory Method 使一个类的实例化延迟到其子类。
- **Flyweight 模式**　运用共享技术有效地支持大量细粒度的对象。
- **Interpreter 模式**　给定一个语言，定义其文法（文法是语言编译分支中的一个术语）的一种表示，并定义一个解释器，该解释器使用该文法表示来解释语言中的句子。
- **Iterator 模式**　提供一种方法顺序访问一个聚合对象中各个元素，而又不需暴露该对象的内部表示。
- **Mediator 模式**　用一个中介对象来封装一系列的对象交互。中介者使各对象不需要显式地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互。
- **Memento 模式**　在不破坏封装性的前提下捕获一个对象的内部状态，并在该对象之外保存这个状态。这样以后就可将该对象恢复到保存的状态。
- **Observer 模式**　定义对象间的一种一对多的依赖关系，以便当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动刷新。

- **Prototype 模式** 用原型实例指定创建对象的种类，并且通过复制这个原型来创建新的对象。
- **Proxy 模式** 为其他对象提供一个代理以控制对这个对象的访问。
- **Singleton 模式** 保证一个类仅有一个实例，并提供一个访问它的全局访问点。
- **State 模式** 允许一个对象在其内部状态改变时改变其行为。对象看起来似乎修改了其所属的类。
- **Strategy 模式** 定义一系列的算法，将其一个个封装起来，并且使它们可相互替换。本模式使得算法的变化可独立于使用它的客户。
- **Template Method 模式** 定义一个操作中的算法的骨架，而将一些步骤延迟到子类中。Template Method 使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。
- **Visitor 模式** 表示一个作用于某对象结构中各元素的操作。可以在不改变各元素的类的前提下定义作用于这些元素的新操作。

设计模式通常根据一些公共特性而组合在一起，GOF 将设计模式划分为 3 类：创建型（Creational）、行为型（Behavioral）和结构型（Structural）。表 16-1 所示是按这种方法分类上述模式的列表。

表 16-1　GOF 设计模式分类表

目的 / 范围	创建型	结构型	行为型
类	Factory Method	Adapter（类）	Interpreter Template Method
对象	Abstract Factory Builder Prototype Singleton	Adapter（对象） Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

该表根据两条准则对模式进行分类。第一条是目的准则，即模式是用来完成什么工作。模式依据其目的可以分为创建型、结构型和行为型 3 种。创建型模式与对象的创建有关；结构型模式处理类或对象的组合关系；行为型模式对类或对象怎样交互，以及怎样分配职责进行描述。

第二条是范围准则，指定模式主要是用于类还是用于对象。类模式用来处理类和子类之间的关系，这些关系通过继承建立，是静态的，在编译时就已经确定。对象模式处理对象间的关系，这些关系在运行时刻是可以变化的，更具有动态性。从某种意义上说，几乎所有类模式都是使用继承机制。所有“类模式”只指那些集中于处理类之间关系的模式，而大部分模式都属于对象模式的范畴。

16.2　常用设计模式

Ajax 之所以能超越 DHTML，不仅在于其组合这些技术的方式不同，而且在于它使用这些技术的规模不同。相比传统 Web 应用来说它不仅需要更多的 JavaScript，而且代码在浏览器中驻留的时间比较长。因此 Ajax 需要以一种与传统 Web 开发所不同的方式来管理这些复杂的代码。管理这些复杂代码的有效方法之一就是在 Ajax 应用中使用恰当的设计模式。

在 Ajax 开发中常用的设计模式主要有 5 种设计模式：Facade 模式、Adaptor 模式、Observer 模式、Command 模式和 MVC 模式。

16.2.1　Facade 模式

在软件开发系统中，客户程序经常会与复杂系统的内部子系统之间产生耦合，而导致客户程序随着子系统的变化而变化。那么如何简化客户程序与子系统之间的交互接口、如何将复杂系统的内部子系统与客户程序之间的依赖解耦，这就是要说的 Facade 模式。Facade 模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。

Facade 模式的作用主要有两点。

- 为一个复杂的子系统（Sub System）提供一个统一的、简单的接口供调用方使用，有益于简化调用方的工作。
- 降低系统的耦合度。一批事物的变化率远大于一个事物。应用 Facade 模式的情景，一般是由于子系统太复杂。

例如，浏览器厂商提供非标准机制来获取 XMLHttpRequest 对象，用于发送异步请求到服务器。IE 浏览器通过访问 ActiveX 组件来获取这个对象，而 Mozilla 则是以内置对象的形式提供这个对象。

在开发 Web 程序时完成检测浏览器版本的代码都是类似的，可以通过一个函数来隐藏这些代码，以便使其他代码更加突出、容易阅读。这就是重构的一个基本原则——不要重复的代码。

从设计模式的角度看，这就是 Facade 模式。Facade 模式用来为一个服务或一些功能的不同实现方式提供公共的访问点。例如，XMLHttpRequest 对象提供了服务器与客户浏览器之间的异步通信，应用 Facade 模式后通过对象的方法函数隐藏其创建方法，这样应用就不需要考虑其如何实现的，如图 16-1 所示。

当调用方需要执行某些任务时，需要执行一连串操作。而随着子系统的升级，这一连串操作的变化可能性非常大。但是不管这个子系统如何变化，这一连串操作所提供的功能或者意义永远不变。所以 Facade 模式有利于降低耦合度。接着上面的示例使用 Facade 模式来简化对 XMLHttpRequest 对象的调用过程。

当需要用 XMLHttpRequest 对象向服务器请求数据时，仅仅是为了请求一次数据就需要 5 个步骤。

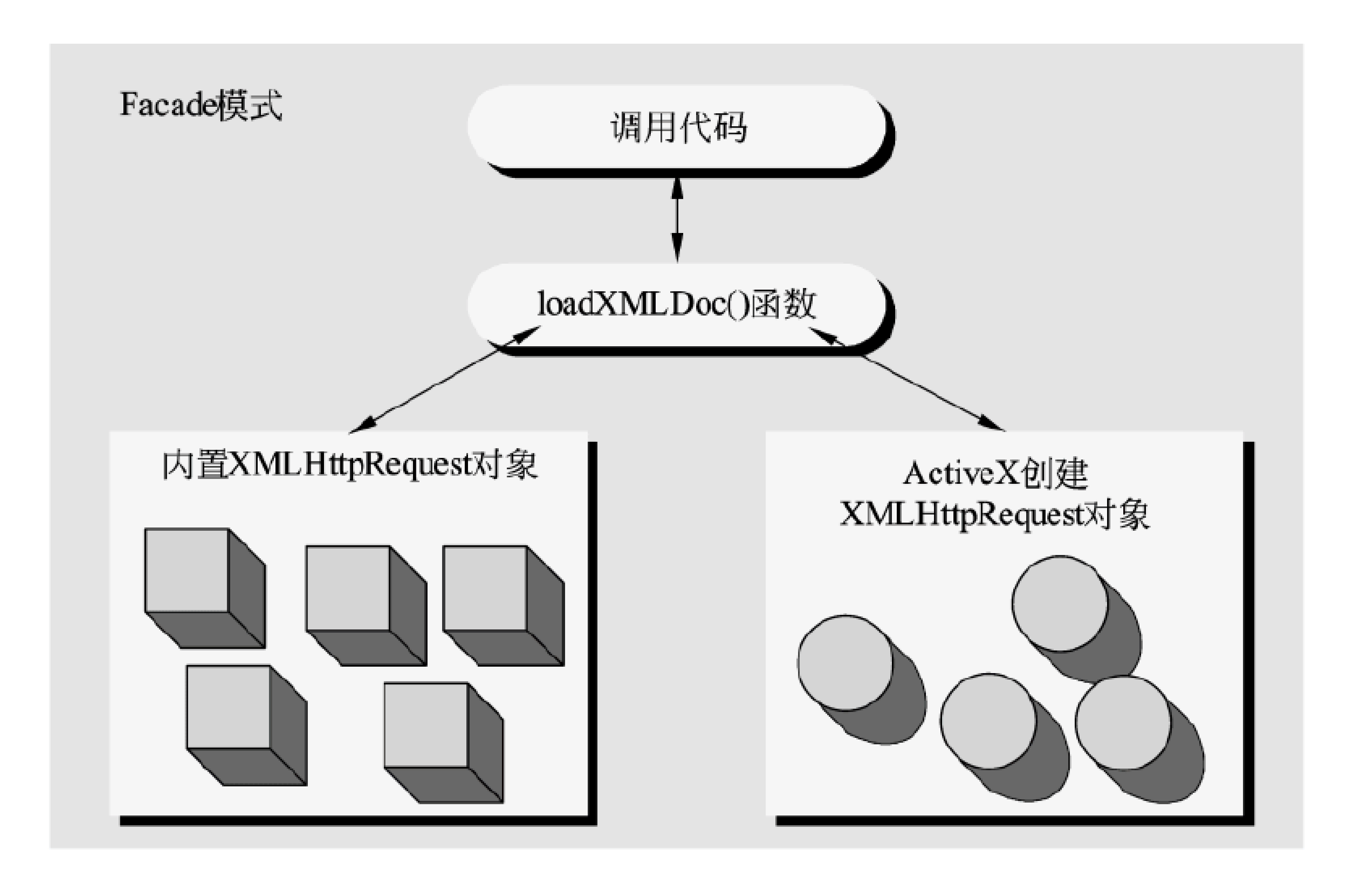


图 16-1　Facade 模式示意图

（1）建立 XMLHttpRequest 对象。
（2）注册 callback()函数。
（3）用 open()方法设置请求方式、URL 地址和请求模式。
（4）用 send()方法发送请求。
（5）监视请求的状态，并对相应的状态进行响应处理。

在本章之前使用的 createXMLHttpRequest()方法封装了创建 XMLHttpRequest 对象的过程，很好地屏蔽了创建对象的复杂性。当需要从服务器端获得数据时，关心的是数据而不是在于服务器交互的过程，因此需要一个函数来隐藏这一系列操作。Facade 模式可以很好的完成这一任务。用来隐藏这一系列操作的对象为 Message，该对象的源代码如代码 16.1 所示。

代码 16.1　Message 对象代码

```
var xhr;
var refreshDiv;
function Message()
{
    this.xhr = createXMLHttpRequest();
}
Message.prototype.sendMessage = function (url,refreshDivId)
{
  this.xhr.open('post',url);
    refreshDiv = refreshDivId;
    xhr = this.xhr;
    refreshDiv = refreshDivId;
  this.xhr.onreadystatechange = this.stateChange;
  this.xhr.send(null);
}
```

```
Message.prototype.stateChange = function()
{
    alert(xhr.readystate);
    if (xhr.readyState == 4)
     {
         alert(xhr.status);
        if (xhr.status == 200)
        {
          document.getElementById(refreshDiv).innerHTML = xhr.responseText;
        }
     }
}
```

该文件保存为 example.js，如果未使用 Facade 模式，则发送异步请求需要在客户端重复繁琐的调用工作。使用 Facade 模式后，通过该模式提供的接口就可以很简单方便地实现发送异步请求的工作。在该文件中 Message 对象就很好地屏蔽发送请求的复杂性。当需要处理服务器响应时，只需要如下两行代码就可以轻松处理，代码如下所示：

```
var message = new Message();
message.sendMessage('t.xml','info');
```

由上可以看到，实际上 Message 类就是一个简单的接口，它将整个复杂的接口简单化、统一化。Facade 模式注重的是简化接口，更多的时候是从架构的层次去看整个系统，而并非单个类的层次。主要适用于实现如下效果。

- Facade 模式对客户屏蔽了子系统组件，因而减少了客户处理对象的数目并使得子系统使用起来更加方便。
- Facade 模式实现了子系统与客户之间的松耦合关系，而子系统内部的功能组件往往是紧耦合的。松耦合关系使得子系统的组件变化不会影响到它的客户。
- 如果应用需要，并不限制它们使用子系统类。因此可以在系统易用性与通用性之间选择。

16.2.2　Adapter 模式

Adapter 模式解决接口不一致的问题。在实际的应用程序中，有时客户端（这里指调用方）想要调用的接口与实际上服务端（这里指被调用方）所提供的接口不一致。出现这种情况可能会有两种选择：一种是修改调用方或者被调用方的接口，使之互相适应；另一种就是在调用方和被调用方之间加入一个适配器（Adapter），让其隐藏两者之间的差异。

在 Adapter 模式里，Adapter 对象所起的作用就是一个接口适配器。一个 Adapter 类会实现（Implements）调用方所期待的接口，并且在类中通过委派（Delegate）来调用被调用方所提供的服务，从而实现两种不同接口的连接。

Adapter 模式分为两种实现方式：对象适配器（Object Adapters）和类适配器（Class Adapters）。其中对象适配器（Object Adapters）通过组合（composition）实现，而类适配器（Class

Adapters）通过多继承实现。

在 Ajax 应用中经常用到的 XMLHttpRequest 对象，就是需要 Adapter 模式的一个很好的实例。XMLHttpRequest 对象非 W3C 标准，所以尽管现在有较新的浏览器都支持 XMLHttpRequest 对象，但其具体实现是不一致的。在微软的 IE 里 XMLHttpRequest 是以 ActiveXObject 的样式实现的，而在 Mozilla 浏览器里其又以一种 build-in 对象的形式实现。

对于常用的应用程序来说，并不关心这些实现细节，而是如何能够获得一个可以供使用 XMLHttpRequest 对象。获得一个可用 XMLHttpRequest 对象的代码如代码 16.2 所示。

代码 16.2　创建 XMLHttpRequest 对象的 createXMLHttpRequest()函数

```
function createXMLHttpRequest()
{
  var XHR = false;
  if (window.XMLHttpRequest)
  {
    //创建非 IE 浏览器中的 XMLHttpRequest 对象
    XHR = new XMLHttpRequest();
  }
  else if (window.ActiveXObject)
  {
    //创建 IE 浏览器中的 XMLHttpRequest 对象
    try
    {
       XHR = new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch (e1)
    {
      try
      {
        XHR = new ActiveXObject("Microsoft.XMLHTTP");
      }
      catch (e2)
      {
        // 不能创建 XMLHttpRequest 对象
         alert('不能创建异步请求对象!');
      }
    }
  }
  return XHR;
}
```

该文件保存在 example1.js 文件中，该文件中的 try…catch 语句实现了跨平台性，createXMLHttpRequest()函数封装了与特定浏览器相关的创建 XMLHttpRequest 对象的方法，该函数便是一个 Adapter。通过适配器来适应各种平台的变化。程序代码期待一个创建

XMLHttpRequest 的统一接口，这段代码实现了这个接口，并且通过委托的机制自动用各种方法在各种不同的平台下实例化一个 XMLHttpRequest 对象。

16.2.3　Observer 模式

Observer 模式作为行为模式中的一种，其与众不同的是它的关注重心不是对象的行为，而是两个或多个相互协作类之间的依赖关系。它之所以被称为行为模式，原因是它通过了某种行为来控制这种依赖关系并产生消息通知，进而达到修改被依赖的类的行为或状态的目的。

Observer 模式中最关键的一点是需要对观察者（Observer）角色进行抽象，唯有如此才能解除观察者与被观察者之间的依赖关系。另外被观察者即主体（Subject）角色还需要维护一个集合对象，它是一个观察者对象的列表集合。在事件通知到来时，被观察者将遍历该集合内的观察者对象并调用其相关方法，如图 16-2 所示。Observer 模式在程序设计中使用最为广泛的是事件响应机制。

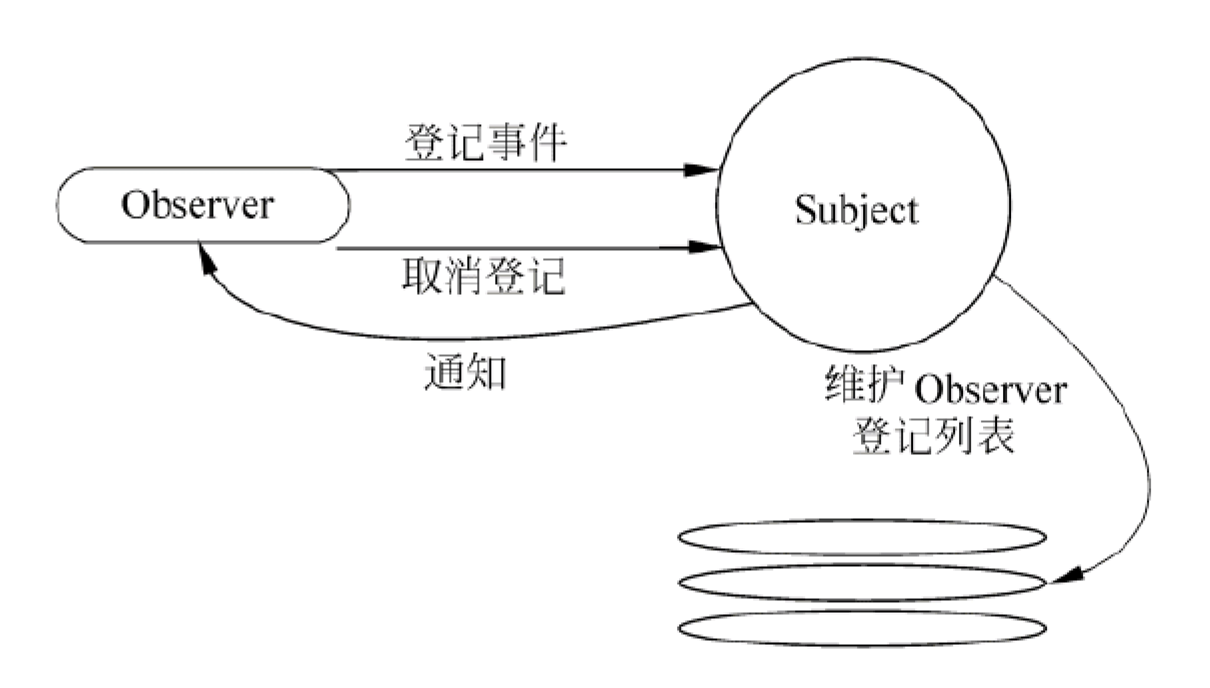


图 16-2　Observer 模式

在 Ajax 应用中，Observer（观察者）模式是事件处理函数。通过 Observer 模式将一对多对象之间的通知依赖关系变得更为松散，提高了程序的可维护性和可扩展性，也很好地符合了开放-封闭原则。综上所述，Observer 模式主要适用于如下情况。

- 当一个抽象模型有两个方面，其中一个方面依赖于另一方面。将这两者封装在独立的对象中以使其可以各自独立地改变和复用。
- 当对一个对象的改变需要同时改变其他对象，而不知道具体有多少对象有待改变。

当一个对象必须通知其他对象，而它又不能假定其他对象是谁。换言之，不希望这些对象是紧密耦合的。

JavaScript 的用户界面严重依赖于事件驱动，而在 Ajax 引入异步请求之后，应用程序需要处理的回调函数和事件也随之增多了。在相对简单的应用中，可以使用某个函数来处理类似鼠标点击或者服务器端数据到达的事件。然而在应用规模和复杂性日渐增长时，可能想要通知几个不同的子系统，这就需要提供一种机制，让对事件感兴趣的各方可以自行登记所需要的事件通知。

例如，当使用 JavaScript 对 DOM 节点进行脚本处理时，通常还需要定义一个 window.onload 函数，这个函数会在页面全部加载完成时执行。当页面加载完成后，页面上将有一个用来显

示从服务器端获取数据的 DOM 元素。如：

```
window.onload=function(){…}
```

假设现在要增加另一个视觉效果，并且也定义了一个 window.onload 事件处理函数：

```
window.onload=function(){…}
```

如果分别在独立的页面中测试这两段代码，可以发现它们都可以正常运行。但是将这两段代码放在一起时，第二个 window.onload 函数就会覆盖第一个，导致第一个 window.onload 函数调用失败。这个问题就在于在 window 对象上只允许附加一个 onload 事件函数。对于当前这个示例而言，通过将两者组合在一个函数中可以解决这个问题，但是这导致本来毫不相干的代码混杂在了一起。如果要处理的不是两个系统，而是 10 个甚至更多，它们都需要得到对几个 DOM 元素的引用，那么像这样的组合事件处理函数将会变得难以维护，导致从中插入和取出单独的组件将会变得非常困难而且极易出错。

如果应用 Observer 模式设计程序，则可以允许这些子系统通过传递一个函数来登记，从而在发生 onload 事件时得到通知，这些函数会在 window.onload 事件触发时调用。这里是一个简单的实现：

```
window.onloadListeners=new Array();
window.addOnLoadListener(listener){
    window.onloadListeners[window.onloadListeners.length]=listener;
}
```

现在当窗口完全加载完成后，window 对象只需要遍历这个数组，并且依次调用其中的每个方法即可：

```
window.onload=function(){
    for(var i=0;i<window.onloadListeners.length;i++){
    var func=window.onlloadListeners[i];
    func.call();
  }
}
```

在这里 Observer 模式定义的 Subject 对象是内置的 window 对象，一组 Observer 或 Listener 可以将自己登记在这个对象上。

如果每一个子系统都使用这种方法，就可以提供更清晰的方式来设置所有的子系统，而不必将其混杂在一起。还有一点值得指出的是新的 W3C 事件模型也实现了一个多事件处理函数的系统。这里之所以选择在 JavaScript 事件模型之上建造系统，是因为 W3C 的模型在不同浏览器中的实现不一致。

16.2.4 Command 模式

在众多的设计模式中，Command 模式是很简单也很优雅的一种设计模式。Command 模式封装的是命令，将命令发出者的责任和命令执行者的责任分开。

Command 模式将一个请求封装为一个对象，从而可以使用不同的请求对客户进行参数化。例如，用餐时的账单就是 Command 模式的一个例子。服务员接受顾客的订单，把它记在账单上封装，这个订单被排队等待烹饪。注意这里的“账单”是不依赖于菜单的，它可以被不同的顾客使用，因此它可以添入不同的菜单项目。

在下面的情况下应当考虑使用 Command 模式。

- 使用命令模式作为 CallBack 在面向对象系统中的替代。CallBack 讲的便是先将一个函数登记上，然后在以后调用此函数。
- 需要在不同的时间指定请求、将请求排队。一个命令对象和原先的请求发出者可以有不同的生命期。换言之，原先的请求发出者可能已经不在，而命令对象本身仍然是活动的。这时命令的接收者可以是在本地，也可以在网络的另外一个地址。命令对象可以在串行化之后传送到另外一台计算机上。
- 系统需要支持命令的撤销（undo）。命令对象可以将状态存储起来，等到客户端需要撤销命令所产生的效果时，可以调用 undo()方法将命令所产生的效果撤销掉。命令对象还可以提供 redo()方法，以供客户端再需要时再重新实施命令效果。
- 如果一个系统要将系统中所有的数据更新到日志里，以便在系统崩溃时可以根据日志读回所有的数据更新命令，重新调用 Execute()方法一条一条执行这些命令，从而恢复系统在崩溃前所做的数据更新。

综上所述，Command 模式的根本目的在于将“行为请求者”与“行为实现者”解耦，在面向对象语言中，常见的实现手段是“将行为抽象为对象”。Command 模式以面向对象中的“接口-实现”来定义行为接口规范，更严格、更符合抽象原则。

16.2.5　MVC 模式

MVC 是一种目前广泛流行的软件设计模式，早在 20 世纪 70 年代 IBM 就开始对 MVC 设计模式的研究。近来随着 J2EE 的成熟，它正在成为在 J2EE 平台上推荐的一种设计模型，也是广大 Java 开发者非常感兴趣的设计模型。MVC 模式也逐渐在 PHP 和 ColdFusion 开发者中运用，并有增长趋势。随着网络应用的快速增加，MVC 模式对于 Web 应用的开发无疑是一种非常先进的设计思想。无论选择哪种语言，无论应用多么复杂，它为理解分析应用模型时提供最基本的分析方法、为产品提供清晰的设计框架、为软件工程提供规范的依据。

1. MVC 核心组件

MVC（Model-View-Controller，模型-视图-控制器）模式的目标是将一个应用的输入、处理、输出流程按照模型层（Model）、视图层（View）和控制层（Controller）的方式进行分离。

- **模型层（Model）**

模型就是业务流程/状态的处理以及业务规则的制定。业务流程的处理过程对其他层来说是黑箱操作，模型接受视图请求的数据并返回最终的处理结果。业务模型的设计可以说是 MVC 最主要的核心。它从应用技术实现的角度对模型做了进一步的划分，以便充分利用现有的组件，但它不能作为应用设计模型的框架。它仅仅告诉按这种模型设计就可以利用某些技术组件，从而减少技术上的困难。

业务模型还有一个很重要的模型那就是数据模型。数据模型主要指实体对象的数据保存（持久化）。例如，将一张订单保存到数据库，从数据库获取订单。可以将这个模型单独列出，所有有关数据库的操作只限制在该模型中。

- **视图层（View）**

视图代表用户交互界面，对于 Web 应用来说可以概括为 HTML 界面。应用的复杂性和规模性使得界面的处理也变得具有挑战性。一个应用可能有很多不同的视图，MVC 设计模式对于视图的处理仅限于视图上数据的采集和处理以及用户的请求，而不包括在视图上的业务流程的处理。业务流程的处理交给模型（Model）处理。例如，一个订单的视图只接受来自模型的数据并显示给用户，以及将用户界面的输入数据和请求传递给控制和模型。

- **控制层（Controller）**

控制可以理解为从用户接收请求，将模型与视图匹配在一起，共同完成用户的请求。划分控制层的作用也很明显，控制器实质就是一个分发器，选择什么样的模型，选择什么样的视图，可以完成什么样的用户请求。控制层并不做任何的数据处理。例如，用户单击一个链接，控制层接受请求后并不处理业务信息，它只将用户的信息传递给模型，告诉模型做什么，选择符合要求的视图返回给用户。因此一个模型可能对应多个视图，一个视图也可能对应多个模型。

2. MVC 的优点

在开发传统的 Web 应用中，由于很少使用设计模式或框架，所以开发速度往往比较快。但由于数据页面的分离不是很直接，因而很难体现出业务模型的样子或者模型的重用性。产品设计弹性力度很小，很难满足用户的变化性需求。MVC 要求对应用分层，虽然要花费额外的工作，但产品的结构清晰，产品的应用通过模型可以得到更好地体现，这为需求变更和后期维护提供了很好的支持。

首先最重要的是应该有多个视图对应一个模型的能力。在目前用户需求的快速变化下，可能有多种方式访问应用的要求。例如，订单模型可能有本系统的订单，也有网上订单，或者其他系统的订单，但对于订单的处理都是一样，也就是说订单的处理是一致的。按 MVC 设计模式，一个订单模型以及多个视图即可解决问题。这样减少了代码的复制，即减少了代码的维护量。一旦模型发生改变，也易于维护，如图 16-3 所示。

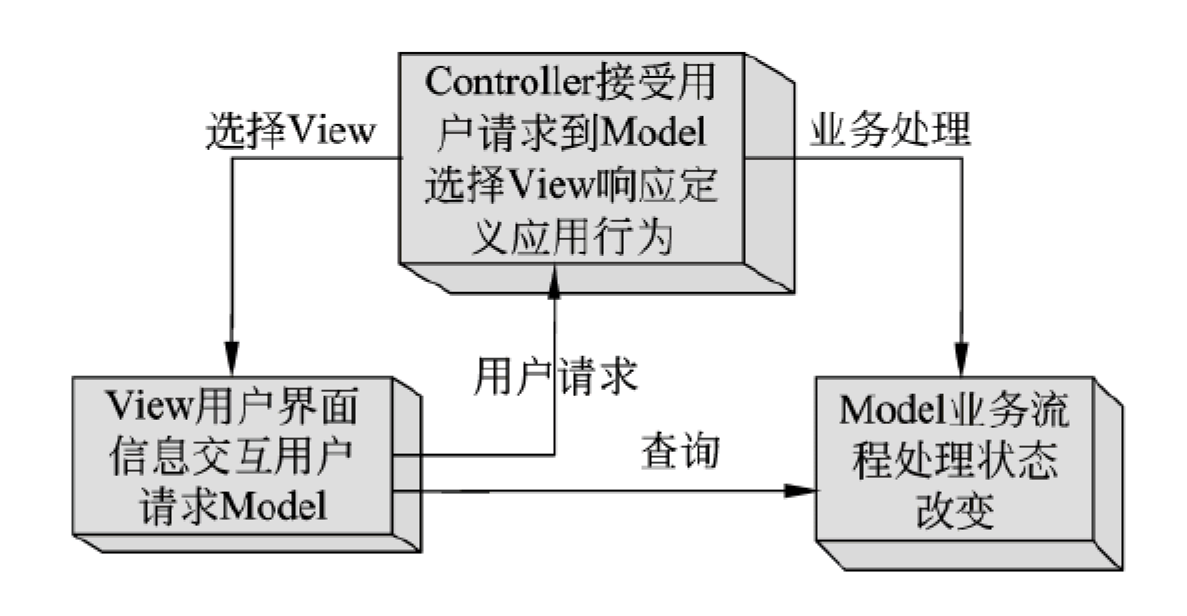


图 16-3　MVC 设计模型

此外由于模型返回的数据不带任何显示格式，因而这些模型也可直接应用于接口的使用。

其次由于一个应用被分离为 3 层，因此有时改变其中的一层就能满足应用的改变。一个应用的业务流程或者业务规则的改变只需改动 MVC 的模型层。控制层的概念也很有效，由于它将不同的模型和不同的视图组合在一起完成不同的请求，因此控制层可以说是包含了用户请求权限的概念。

最后它还有利于软件工程化管理。由于不同的层各司其职，每一层不同的应用具有某些相同的特征，有利于通过工程化、工具化产生管理程序代码。

3. MVC 的缺点

MVC 的设计实现并不十分容易，理解起来比较容易，但对开发人员的要求比较高。MVC 只是一种基本的设计思想，还需要详细的设计规划。

模型和视图的严格分离可能使得调试困难一些，但错误比较明显容易发现。经验表明，MVC 由于将应用分为 3 层，意味着代码文件增多，因此对于文件的管理需要费点心思。

综合上述，MVC 是构筑软件非常好的基本模式，至少将业务处理与显示分离。强迫将应用分为模型、视图以及控制层，使得开发者会认真考虑应用的额外复杂性，将这些想法融进到架构中增加了应用的可拓展性。如果能把握到这一点，MVC 模式会使得开发的应用更加强壮、更加有弹性、更加个性化。

16.3　应用 MVC 模式

在一个典型的 Ajax 应用中至少有 3 层，每一层在应用的生命周期中扮演不同的角色，它们都有助于开发出清晰、组织良好的代码。图 16-4 演示了这些不同规模的 MVC 模式如何嵌套在应用的架构中。

图 16-4 演示在嵌套的 MVC 架构中 MVC 模式以不同的规模重复自己。从最外层的级别可以看到 MVC 模式作为整体定义了应用的工作流，模型位于 Web 服务器端。在较小的规模中，MVC 模式在客户端应用中也被重复使用。

图 16-4　嵌套的 MVC 模式

16.3.1 应用 Ajax 视图

职责分离，相互不影响，无论是页面设计师、图形处理人员，还是编码人员都希望如此。好的设计模式会使后期维护变得轻松，只需少量的工作就可达到事半功倍的效果。即使在开发阶段好的设计模式也使各个部分相互不干扰，通过 API 相互通信，降低了耦合性。当需求改变时，在 API 不变的情况下只需改变相应的部分而不需要更改其他的部分。

视图是可视的页面，由 DOM 元素组成。这些 DOM 元素通过使用 HTML 标记呈现，或者采用编程方式处理。在 MVC 模式中，视图有两个主要的责任：必须为用户提供一个可视的界面以便触发事件，事件用来与控制器对话；也需要在模型改变时做出响应，更新自己，通常需要再次通过控制器进行通信。

在实际的应用中，为了将 Web 页面结构化常将 CSS、HTML 和 JavaScript 定义在分离的文件中。在页面部分，这种分离遵从 MVC：CSS 样式表是视图，HTML/DOM 是模型，并通过简单地将 JavaScript 分离出来并放在一个单独的文件中，可以使页面设计人员和程序员相互隔离，不互相影响。

将所有的 JavaScript 编写在一个分离的文件中是强化视图分离的良好开端。但是即使这样做，如果不注意仍然可能导致视图混入逻辑角色，即充当模型和控制器。如果将 JavaScript 事件处理函数内嵌在页面中，例如：

```
<span class='btnSend' onclick='SendRequest("Get",txtUserName.value);'/>
```

上述代码实际上将无意义的逻辑编写在视图中。因为 SendRequest()函数和它的两个参数 Get 和 txtUserName.value 的含义是什么对于页面设计人员而言是不需要知道的。按照 MVC 的规定，视图和模型不应该直接通信，一种解决方案是使用额外的层来将其分离。

为了保持控制器、模式和视图的分离，可以采用编程方式添加事件。除了使用嵌入的事件处理函数，还可以指定某种类型的记号，它随后会被代码获得。有几种方法来做这个记号。一种方法是为元素附加唯一的 ID，然后以每个元素为基础指定事件处理函数。例如：

```
<div class='btnSend' id='btnSendReq'>
```

下面的代码作为 window.onload 回调的一部分来执行，例如：

```
var dataBtn=document.getElementById('btnSendReq');
dataBtn.onclick=SendRequest;
```

另一种方法是如果希望多个事件处理函数执行相同的操作，则可以在视图中使用不唯一的标识。最简单的方法是定义一个额外的 CSS 类，然后再使用 CSS 间接添加事件。

例如，在下面的示例中将鼠标事件绑定在虚拟的按键上，以模拟计算器的数字按键。下面代码 16.3 中定义了一个包含原始文档结构的简单页面 Calculator.html。

代码 16.3　原始文档：Calculator.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
```

```
<head>
   <title>计算器</title>
    <link type="text/css" rel="stylesheet" href="key.css"/>
    <script src="key.js" type="text/javascript"></script>
    <script type="text/javascript">
       window.onload=assignKeys;
    </script>
</head>
<body>
     <div id="keyboard" class="top Keys"></div>
     <div id="display" class="display"></div>
</body>
</html>
```

在上面的页面中，为容纳键盘的按键定义了一个<div>元素，并且分配了唯一标记属性 ID 和两个样式类。显示按键输入内容则由 id 属性为 display 的<div>元素负责。所使用的 CSS 样式表 key.css 如代码 16.4。

代码 16.4　原始文档的 CSS 样式文件：key.css

```
.Keys{
   background-color:#ffe0d0;
   border:solid maroon 2px;
   overflow:auto;
   margin:4px;
}
.top{
   width:340px;
   height:28px;
   top:0px;
   left:124px;
}
.Button{
   border:solid navy 1px;
   background-color:Fuchsia;
   width:20px;
   height:20px;
   text-align:center;
   position:relative;
   margin:2px;
   float:left;
}
.display{
   font-size:large;
   color:rgb(255,0,0);
   text-align:left;
```

```
}
```

样式 top 定义了按键区的位置属性，按键区的边框和背景属性由 Keys 样式类定义，每个按键属性则由 Button 样式类定义。

文件 key.js 是事件处理函数，其角色为 MVC 中的控制器，其内容如代码 16.5 所示。

代码 16.5　控制器：key.js

```
var notes=new Array("0","1","2","3","4","5","6","7","8","9");//生成的数字按键
function assignKeys(){
        var divs=document.getElementsByTagName("div");   //查找页面的 div 元素
        if(divs){
             for(var i=0;i<divs.length;i++){
                var candidate=divs[i];
                if(candidate.className.indexOf('Keys')>=0)
                                              //查找 class 属性名含 Keys 的 div 元素
                makeKeyBoard(candidate);
              }
        }
 }

function makeKeyBoard(el){  //为<div>元素添加子元素
        for(var i=0;i<notes.length;i++){
            var key=document.createElement("div");
            key.className="Button";  //为子元素添加属性
            key.note=notes[i];
            key.onclick=playNote;
            var value=document.createTextNode(notes[i]);
            key.appendChild(value);
            el.appendChild(key);
          }
 }

function playNote(event){
         var note=this.note;  //获取定制属性
         var display=document.getElementById('display');
         if(note&&display){
            display.innerHTML+=note+", ";
        }
 }
```

当 window.onload 调用 assignKeys()函数时，使用 getElementsByTagName()遍历访问其内部所有的<div>元素。这虽然需要知道一些关于页面结构的知识，但是它允许页面设计人员自由地在页面中将 div 按键以希望的方式任意移动。

16.3.2　应用 Ajax 控制器

在 MVC 设计模式中，控制器的角色是作为模型和视图之间的隔离者。在 Ajax 客户端应用中，控制器层由事件处理函数组成。对于 Web 浏览器而言，现代浏览器支持两种不同的事件模型：传统的事件模型和 W3C 事件模型，并且传统的事件模型正在被新的 W3C 事件模型所取代。

传统的事件模型在 JavaScript 诞生的早期就存在了，它是相当简单和直接的。DOM 元素有几个预先定义的属性，可以赋值为回调函数。例如，在鼠标单击<btnSend>元素时调用 SendRequest()函数，可以使用 onclick 事件：

```
btnSend.onclick=SendRequest;
```

其中 btnSend 是可以通过程序处理的任何 DOM 元素，SendRequest()是一个函数，定义为：

```
function SendRequest (){
    //实现具体功能
}
```

但是在这里要注意当分配事件处理函数时所传递的是一个 Function 对象，而不是对那个对象的调用，因此必须在函数名后面去除圆括号。例如，下面是一个常见的错误：

```
btnSend.onclick=SendRequest();
```

在这种情况下函数将在进行赋值时被调用，而不是当单击 DOM 元素时才被调用。onclick 属性将被设置为函数返回的任何值。

传统事件模型的主要缺点是每个元素只允许有一个事件处理函数。在介绍 Observer 设计模式时，注意到一个被观察者元素在给定时间内可以有任意多个观察者附加在上面。当为 Web 页面编写简单的脚本时，这可能不会成为一个严重的缺点，但是当编写更加复杂的 Ajax 客户端时就可以感觉到更多的限制。

W3C 事件模型要复杂一些，但是可以在一个 DOM 元素上附加任意数目的观察者。在基于 Mozilla 的浏览器中，使用 addEventListener()来添加事件回调函数，使用相应的 removeEventListener()来删除。IE 提供了类似的函数：attachEvent()和 detach-Event()。

这就带来了一个更深层的跨浏览器问题，用户定义的回调处理函数的调用方式有轻微的差异。在基于 Mozilla 的浏览器中，函数被调用时是将收到事件的 DOM 元素作为上下文对象，类似于传统的事件模型一样。而在 IE 中，函数上下文总是 window 对象。因此建议不要使用新的事件模型，虽然传统模型的主要缺点是缺少多个监听器的支持，但是可以通过使用设计模式来解决。

16.3.3　应用 Ajax 模型

模型负责表示应用的业务领域。按照应用的规模来看 DOM 并不是模型，模型是使用

JavaScript 编写的一组代码。像大多数设计模式一样，MVC 的设计思想是基于面向对象的。

JavaScript 并没有设计成一种面向对象语言，虽然用它也可以进行类似于面向对象的方式编程，并且确实可以通过 prototype 机制来定义一些与对象的类非常相似的东西，尤其是一些开发者已经为 JavaScript 实现了继承系统。但是就使用 JavaScript 实现 MVC 而言，还需要将这一模式修改为适应 JavaScript 的编码风格，例如直接作为事件监听器传递 Function 对象。当使用 JavaScript 对象定义模型时，要尽量多地使用这种语言的面向对象开发方法。

下面是一个简单的例子，在本例中将创建一个描述瓶子类型的数据，包括瓶子唯一编号 ID、名称、高度、宽度以及生产日期，以其考察当客户端收到数据时会发生什么。可以定义一个简单的 JavaScript 对象，对应于服务器端定义的 Bottle 对象，如代码 16.6 所示。

代码 16.6　Bottle 对象

```
var Bottle = new Array();
function Bottle(strID, strName, strHeight, strWidth, strMadeDate) {
    this.ID = strID;
    Students[id] = this;
    this.Name = strName;
    this.Height = strHeight;
    this.Width = strWidth;
    this.MadeDate = strMadeDate;
}
```

这里首先定义了一个全局数组，用来保存所有的瓶子信息。这是一个关联数组，瓶子的编号 ID 作为键，保证在同一时间只存在对一瓶子的引用。在构造函数中设置了所有简单的属性。注意：这里没有提供获取或设置方法，也不像一个完整的面向对象语言做的那样支持完全的访问控制——私有、受保护和公有变量及方法。

当解析 XML 数据流时，先创建一个空的 Bottle 对象，然后为每个字段赋值。注意：由于 JavaScript 函数的参数是可变的，所以当调用一个函数时任何缺少值的参数会简单地初始化为 null。所以下面的代码是相同的：

```
var aBottle=new Bottle("CoCo-W-1105-014");
//等价于如下形式
var aBottle=new Bottle("S0372",null,null,null,null);
```

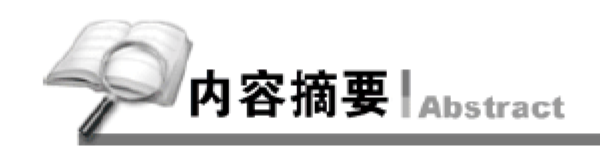
如上述代码所示，必须传递瓶子的编号 ID，以便在构造函数中使用它在全局 BottleList 列表中放置新的对象。

最后重新回顾一下应用 MVC 后的 Ajax 处理过程。首先从数据库创建一个服务器端对象模型。在服务器端可以将数据读入对象，并将其修改，然后保存数据。接下来使用模板系统来从对象模型生成 XML 数据流。最后解析这个数据流从而在 JavaScript 层创建对象模型。当然在客户端也可以修改 JavaScript 模型，然后就这些修改与服务器端模型进行通信。这将迫使开发人员面对两个领域模型副本的问题，以防止它们可能失去同步。

在传统的 Web 应用中，所有的业务逻辑都位于服务器端，所以无论使用什么语言，模型

也位于那里。在 Ajax 应用中，将业务逻辑分布在客户端和服务器端，以便客户端可以在向服务器端发送调用请求之前针对自身做出一些决定。如果客户端仅仅做出非常简单的决定，还可以以编写少量代码的方式来处理，但是这不可能充分利用客户端的逻辑处理，所以仍然会影响系统反应速度。如果授权客户端针对自身做出更加重要的决定，那么它就需要知道一些关于业务领域的事情。现在无法除去服务器端的领域模型，因为一些资源只能在服务器端获得，如持久层的数据库连接等。因此客户端领域模型必须与服务器端的领域模型配合工作。

第17章 PHP常用技巧



通过利用 Ajax 技术，可以提供丰富的、基于浏览器的用户体验。Ajax 技术还可以与数据库技术集成到一起，提供与数据库通信而不刷新浏览器的强有力技术，从而对 Web 应用程序和用户体验产生巨大的影响。本章通过列举基于 Ajax 技术的常见示例详细介绍 PHP+Ajax 技术。例如，基于 Ajax 技术与数据库技术集成的新颖、时尚和技巧性强的示例等。

学习目标 Objective

- 掌握创建工具提示的方法
- 掌握如何使用 HEAD 发送与获取信息，以及请求特点
- 掌握使用 DOM 处理 CSS 代码
- 灵活运用 Ajax 处理流程
- 掌握 PHP+Ajax 的添加、删除和修改操作
- 掌握异步信息的请求和处理
- 掌握 PHP 页面设置缓存
- 熟练掌握 PHP 分页显示数据
- 熟练掌握 PHP 对文件的读写

17.1　创建工具提示

工具提示是 Ajax 中最常用的应用。当浏览一个页面时，如果该页面上有多个选择，用户将鼠标停留在一个给定的选择上就会看到该选择的更多信息。尽管不使用 Ajax 也能达到这个效果，但第一次加载页面时要加载页面上所有信息，其中有大量可能永远也不会用到的信息。而通过使用 Ajax，只会在需要时发送所需信息。下面使用 Ajax 创建一个工具提示的示例，具体开发步骤如下所示。

（1）打开记事本，创建 Demo-1.php 和 Demo-2.php 文件，用来实现发送请求、处理请求和生成响应信息等。

（2）打开 Demo-1.php 文件，首先创建页面显示内容，如代码 17.1 所示。

代码 17.1　Demo-1 页面代码

```
<div id="Layer1">
<img src=img/01.jpg width="100" height="115" id="1" onMouseOver="aa(this)">
  <div id="Layer2">
    <p>出版社：清华大学出版社</p>
    <p>作者：唐晓阳 </p>
    <p>书名：XML 教程</p>
  </div>
</div>
<h3>图书列表</h3>
<p> </p>
<p> </p>
<div id="Layer5"></div>
<p> </p>
 <div id="Layer3"><img src=img/02.jpg width="100" height="115" id="2"
 onMouseOver="aa(this);">
  <div id="Layer4">
    <p>出版社：清华大学出版社</p>
    <p>作者：唐晓阳</p>
    <p>书名：PHP 完全学习手册</p>
  </div>
</div>
```

代码 17.1 创建了 5 个不同的层，其 ID 名称分别为 Layer1、Layer2、Layer3、Layer4 和 Layer5。并且分别为 ID 名称为 1 和 2 的图片设置了鼠标悬浮事件，即当鼠标放置在这些图片上时就会触发鼠标悬浮事件。此时会调用 JavaScript 函数 aa()，在调用 aa()的同时会将 img 作为对象发送给函数。

（3）实现页面请求的发送。在 Demo-1.php 文件中创建 createXMLHttpRequest()和 aa()函数，分别用来实现创建异步传输对象和发送请求，如代码 17.2 所示。

代码 17.2　页面请求的发送

```
<script type=text/javascript>
 var xmlHttp;
       function createXMLHttpRequest(){
         if(window.ActiveXObject){
                xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
            }
         else if(window.xmlHttpRequest){
                xmlHttp=new xmlHttpRequest();
            }
         }
function aa(obj){
  var id=obj.id;
  createXMLHttpRequest();
```

```
    var url="Demo-2.php?key="+escape(obj.id);
    xmlHttp.open("GET",url,true);
    xmlHttp.onreadystatechange=show;
    xmlHttp.send(null);
  }
</script>
```

在函数 aa()中，首先创建变量 id 获取参数 obj 的 id，此时 obj 参数为图片对象，obj.id 表示获取该图片的 ID 号。根据 ID 号不同，而进行不同的操作。下面调用 createXMLHttpRequest()函数创建异步传输对象 xmlHttp，并创建变量 url，其值为服务器端程序 Demo-2.php 以及传递的参数图片 id。然后使用 GET 方法发送一个请求，设定 onreadystatechange 的属性值为 show，即处理服务器度端响应的函数是 show()。

（4）实现服务器端响应的处理。当服务器端返回响应信息后，客户端就会调用 show()函数处理服务器端响应，该函数程序如代码 17.3 所示。

代码 17.3　函数 show()

```
function show(){
 if (xmlHttp.readyState == 4) {
  if (xmlHttp.status == 200) {
     document.getElementById("Layer5").innerHTML = "库存信息:<span style=
     'color:red;font-size:18'>" +xmlHttp.responseText+"</span>";
  }
 }
}
```

在 show()函数中，将从服务器端获取的响应信息 xmlHttp.responseText，赋值给 ID 名称为 Layer5 的层。

（5）实现服务器端程序。服务器端程序主要根据客户端发送的 ID 信息，返回不同的响应信息，如代码 17.4 所示。

代码 17.4　实现服务器端程序

```
<?php
header('Content-type: text/html;charset=GB2312');
$num=$_GET['key'];
if($num==1)
   echo "价格: 66 元。现在有货";
if($num==2)
   echo "价格: 88 元。库存有限，请订货";
?>
```

header()函数首先设定 PHP 页面返回类型以及编码格式。使用$_GET[]获取客户端传递的 ID 值，如果值为 1 输出一个字符串，否则输出另外一个字符串。

（6）部署和运行。将上述两个文件保存到 F:\MyWeb\apache\htdoc\PHP\17.1 目录下。打开 IE 浏览器，在地址栏中输入“http://localhost/PHP/17.1/Demo-1.php”，单击【转到】按钮，页

面显示效果如图 17-1 所示，并且当将鼠标放在不同的图片之上时会在页面右侧显示红色提示信息。



图 17-1　工具提示页面效果图

17.2　读取响应首部

有时可能需要从服务器获取一些内容，例如验证服务器是否正常运行，也许只想读取服务器发出的响应首部而忽略内容。通过读取响应首部可以得出 Content-Type（内容类型）、Content-Length（内容长度），甚至 Last-Modified（最后一次修改）的日期。

如果只关注响应首部，完成这样一个请求的标准做法是 HEAD 请求，而不是前面讨论的 GET 或 POST 请求。当服务器对 HEAD 请求做出响应时，它只发送响应首部而忽略内容，即使可以向浏览器返回请求的内容，也不会真的将内容返回。由于忽略了内容，HEAD 请求的响应比对 GET 或 POST 的响应相对要快得多。下面创建一个示例，具体步骤如下所示。

（1）打开记事本，创建 17.2-2.php 和 readingResponseHeaders.xml 文件，用来实现发送 HEAD 请求、处理响应信息和返回响应信息。

（2）打开 17.2-2.php 文件，首先实现 PHP 页面内容显示，其代码如下所示：

```
<fieldset><legend>读取网页响应头</legend>
<a href="javascript:doHeaderRequest('allResponseHeaders',
'readingResponseHeaders.xml')">读取所有文件头</a><br>
<a href="javascript:doHeaderRequest('lastModified','readingResponseHeaders.
xml')">获取文件修改日期</a><br>
<a href="javascript:doHeaderRequest('isResourceAvailable',
'readingResponseHeaders.xml')">获取现存资源</a><br>
<a href="javascript:doHeaderRequest('isResourceAvailable',
'not-available.xml')">获取不存在资源</a>
</fieldset>
```

上述代码创建了 4 个超级链接，当单击每个超级链接时就会触发 JavaScript 函数

doHeaderRequest()。该函数第一个参数表示要获取具体 HEAD 头信息，第二个参数表示操作的服务器端程序。单击不同的超级链接会向服务器端发送不同的参数。

（3）实现异步传输对象的创建和请求发送。在 17.2-2.php 文件中，如果要获取服务器端文件返回的 HEAD 信息，需要创建相应的 JavaScript 函数 createXMLHttpRequest() 和 doHeaderRequest(request,url)，实现异步传输对象的创建和请求发送。主要脚本程序如代码 17.5 所示。

代码 17.5　脚本程序

```
<script type="text/javascript">
var xmlHttp;
var requestType="";
function createXMLHttpRequest(){
   if(window.ActiveXObject){
      xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
   else if(window.XMLHttpRequest){
      xmlHttp=new XMLHttpRequest();
   }
}
function doHeaderRequest(request,url){
  requestType=request;
  createXMLHttpRequest();
  xmlHttp.onreadystatechange=handleStateChange;
  xmlHttp.open("HEAD",url,true);
  xmlHttp.send(null);
}
</script>
```

在上述代码中，创建变量 xmlHttp 用来获取实现的异步传输对象，变量 requestType 表示 HEAD 的请求类型。在 doHeaderRequest(request,url) 函数中将参数 request 的值赋值给 requestType，并调用 createXMLHttpRequest()实现异步传输对象。下面设置 handleStateChange() 函数为处理服务器端响应信息的函数，open()方法表示以 HEAD 方式发送客户端请求，即只请求服务器端页面的文件头信息，send()方法表示向服务器端发送空信息。

（4）实现响应信息的处理。当服务器端返回响应信息后，客户端就会调用 handleStateChange()函数处理返回信息，并且 handleStateChange()函数会根据服务器端返回的不同信息调用不同处理函数。该步骤程序如代码 17.6 所示。

代码 17.6　响应信息的处理程序

```
function handleStateChange(){
   if(xmlHttp.readyState==4){
      if(requestType=="allResponseHeaders"){
         getAllResponseHeaders();
      }
```

```
            else if(requestType=="lastModified"){
               getLastModified();
            }
            else if(requestType=="isResourceAvailable"){
               getIsResourceAvailable();
            }
         }
      }
      function getAllResponseHeaders(){
          alert(xmlHttp.getAllResponseHeaders());
      }
      function getLastModified(){
         alert("Last Modified:"+xmlHttp.getResponseHeader("Last-Modified"));
      }
      function getIsResourceAvailable(){
         if(xmlHttp.status==200){
            alert("Successful response");
         }
         else if(xmlHttp.status==404){
            alert("Resource is unavailable");
         }
      }
```

在 handleStateChange()函数中，如果请求的类型为 allResponseHeaders，即获取服务器端文件所有的头信息，此时调用 JavaScript 函数 getAllResponseHeaders()。在 getAllResponse-Headers()函数中，调用异步传输对象 xmlHttp 的 getAllResponseHeaders()函数以对话框形式显示返回信息。如果请求类型为 lastModified，即获取服务器端文件的修改时间，则调用函数 getLastModified()。在 getLastModified()函数中，使用 getResponseHeader("Last-Modified")返回响应信息。

如果响应类型为 isResourceAvailable，则调用函数 getIsResourceAvailable()。在函数 getIsResourceAvailable()中，需要检查服务器返回 HTTP 状态码。XMLHttpRequest 对象的 status 属性将 HTTP 状态作为一个整数返回。如果状态码为 200，指示这是正常的成功服务器响应；如果服务器状态是 404，表示请求的资源并不存在。

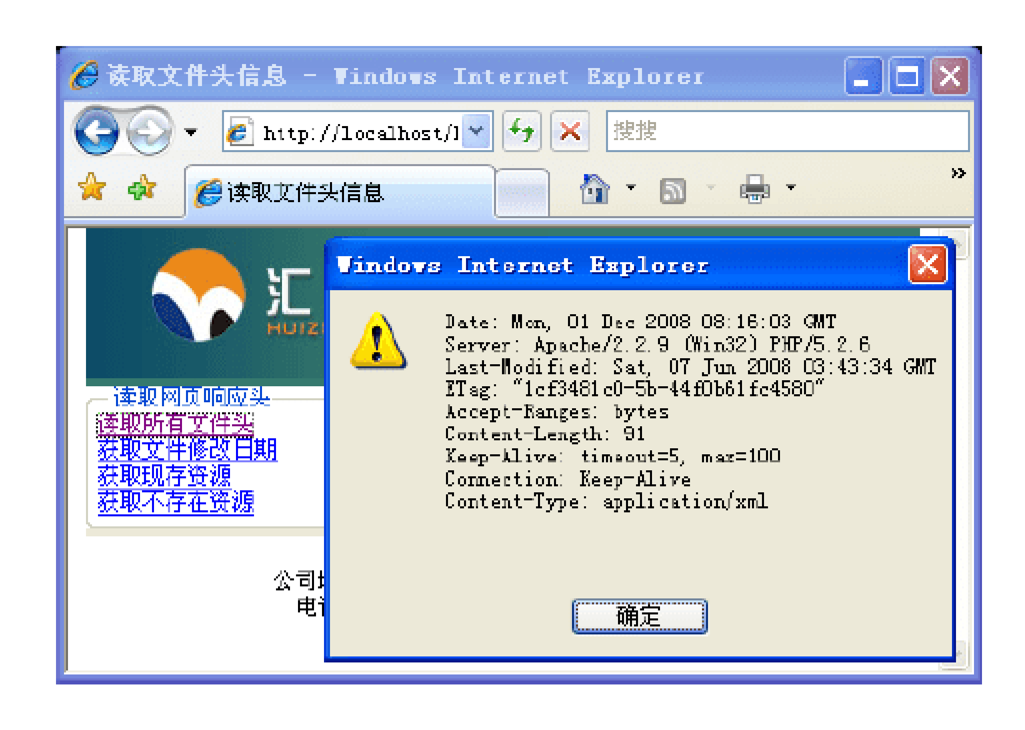
（5）服务器端文件。本示例只是获取服务器端文件的头，所以服务器端文件没有相关的数据和复杂的操作，可以是一个空文件。打开 readingResponseHeaders.xml 文件，其代码如下所示：

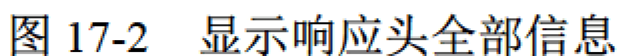
```
<?xml version="1.0" encoding="UTF-8"?>
<readingResponseHeaders></readingResponseHeaders>
```

在保存该文件时需要将此文件编码格式设为 UTF-8。

（6）部署和运行。将上述两个文件保存到 F:\MyWeb\apache\htdocs\PHP\17.2 目录下。打开 IE 浏览器，在地址栏中输入“http://localhost/PHP/17.2/17.2-2.php”，单击【转到】按钮，会显示相应页面。在页面中单击【读取所有文件头】超级链接，页面显示效果如图 17-2 所示。

单击【获取文件修改日期】超级链接，页面显示效果如图 17-3 所示。单击【获取现存资源】超级链接，页面显示效果如图 17-4 所示。




图 17-2　显示响应头全部信息

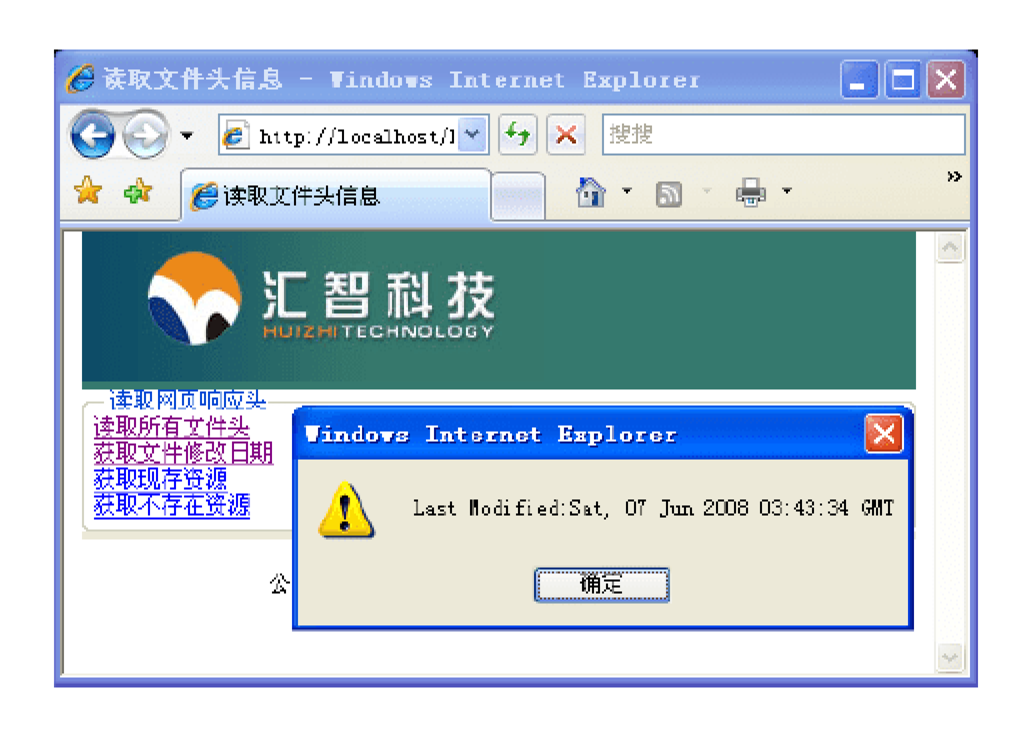


图 17-3　显示文件修改时间

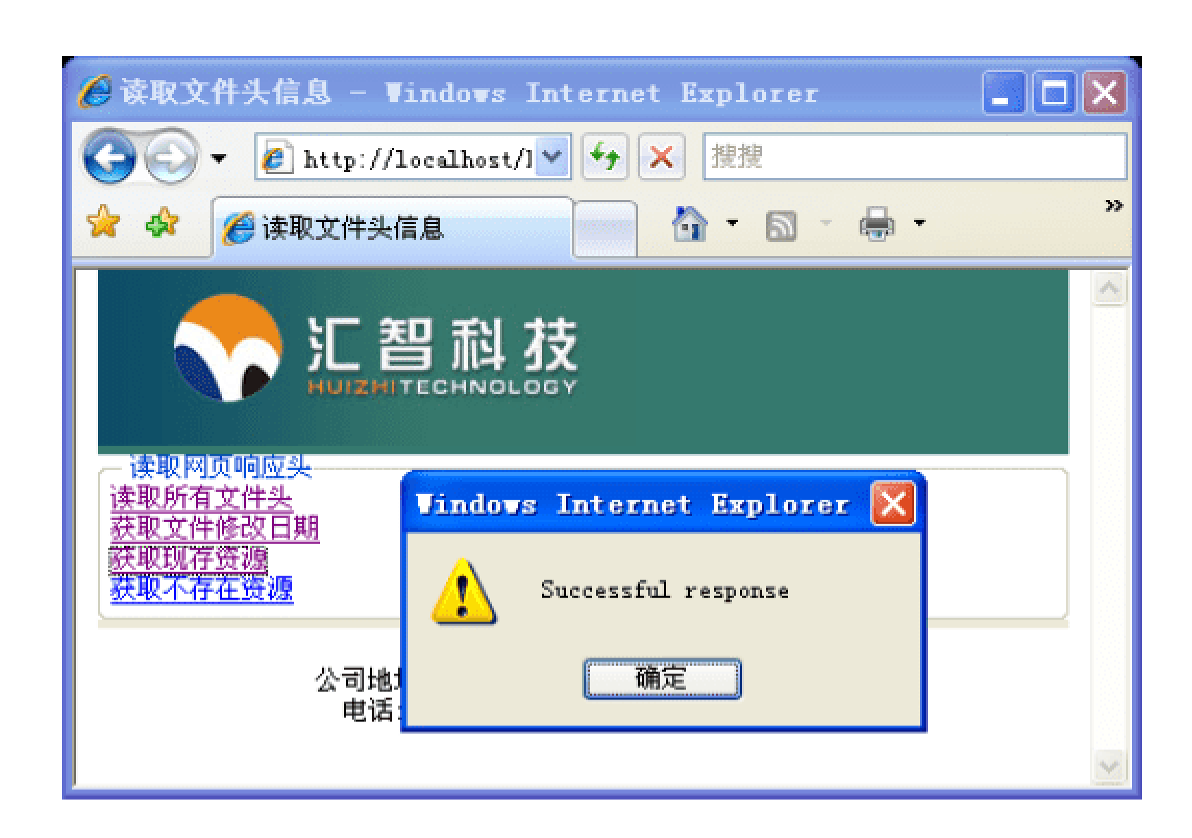


图 17-4　获取现存资源显示

17.3　显示进度条

通常在程序要长时间运行一个事务（操作）时会给用户一些提示，以确保用户能很容易地看到系统的状态。在 Ajax 之前，要在 Web 应用中做到这一点很不容易。而如果是一个富客户端应用，只需显示一个进度条就能使用户知道目前所处状况。

下面使用 Ajax 技术创建一个为 Web 应用建立进度条的示例，具体开发步骤如下所示。

（1）打开记事本，创建 17.3-1.php 和 17.3-2.php 文件，用来实现客户端程序和服务器端程序。

（2）打开 17.3-1.php 文件，为 PHP 页面内容布局。页面布局程序如代码 17.7 所示。

代码 17.7　页面布局代码

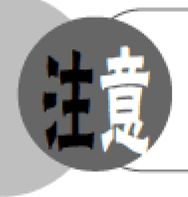
```
<fieldset>
  <legend>Ajax 进度条示例</legend>
  运行进度条:<input type="button" id="go" value="运行"  onclick="go()"/>
```

```
     <p></p>
     <table align="center">
    <tbody>
           <tr><td>
             <div id="progressBar" style="padding:2px;border:solid black 2px;
             visibility?hidden">
                <span id="block1">   </span>
                <span id="block2">   </span>
                <span id="block3">   </span>
                <span id="block4">   </span>
                <span id="block5">   </span>
                <span id="block6">   </span>
                <span id="block7">   </span>
                <span id="block8">   </span>
                <span id="block9">   </span>
             </div></td>
           </tr>
           <tr><td align="center" id="complete"></td> </tr>
    </tbody></table>
   </fieldset>
```

在上述代码中，创建了一个表单元素按钮，其 ID 名称为 go，当单击此按钮时会调用 go() 函数。接下来创建了一个 HTML 表格，在第一个单元格中嵌入了一个层，其 ID 名称为 progressBar，在该层的上面创建了 9 个 span 元素，其 ID 名称依次为 block1、block2、block3、…等。

（3）设置页面编码和页面不允许缓存。17.3-1.php 页面如果需要不断地更新，即显示不同的 span 元素，需要本页面不断地进行发送请求并刷新，故需要设置本页面不允许缓存。其代码如下所示：

```
<?php
header('Content-type: text/html;charset=GB2312');
header("cache-control:no-cache,must-revalidate");
?>
```



上述代码需要放到文件的开始位置。

（4）实现第一次发送请求和处理请求。因为本示例实现的原理是根据不同的服务器响应信息，即返回值，依次显示 span 块，从而造成视觉上的向前运行。当单击按钮，进度条开始显示时，需要向服务器端发送请求，并依据所获取的响应信息显示指定的 span 元素。该步骤程序如代码 17.8 所示。

代码 17.8　显示指定的 span 元素代码

```
<script type="text/javascript">
```

```
var xmlHttp;                          //声明 xmlHttpRequest 对象
var key;
var bar_color='6C8ACC';               //进度条颜色
var span_id='block';                  //设置 span 元素
var clear="   "        //设置空白
var percent_complete;
function createXMLHttpRequest(){      //创建 xmlHttpRequest 对象
      if (window.ActiveXObject){
       xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
      else if(window.XMLHttpRequest){
             xmlHttp = new XMLHttpRequest();
            }
     }
     function go(){                   //执行发送请求方法
            createXMLHttpRequest();
            checkDiv();               //提示层信息
            var url = "17.3-2.php?task=create";
            var button = document.getElementById("go");    //隐藏按钮
            button.disabled = true;
            xmlHttp.open("GET",url,true);
            xmlHttp.onreadystatechange = goCallback;
            xmlHttp.send(null);
     }
     function goCallback(){           //回调方法
      if (xmlHttp.readyState == 4){
       if (xmlHttp.status == 200){
             setTimeout("pollServer()",2000);              //设置更新时间
                   }
            }
     }
     function checkDiv(){             //检查提示层信息
      var progress_bar = document.getElementById("progressBar");
            if (progress_bar.style.visibility == "visible"){
                   clearBar();    //清除进度条
                   document.getElementById("complete").innerHTML = "";
            }else{
             progress_bar.style.visibility = "visible";
            }
     }
     function clearBar(){             //清除进度条
      for (var i = 1; i < 10; i++){
                   var elem = document.getElementById("block" + i);
                  elem.innerHTML = clear;
                  elem.style.backgroundColor = "white";
```

```
                }
        }
    </script>
```

在代码 17.8 中，当用户单击【运行】按钮时触发 go()函数。在 go()函数中，首先调用 createXMLHttpRequest()函数创建异步传输对象，并调用检查提示层信息的 checkDiv()函数。接下来创建变量 url，其值为服务器端文件 17.3-2.php 以及参数 task。然后获取 ID 名称为 go 的按钮对象，使用语句“button.disabled = true”设置按钮失效。最后使用 open()方法向服务器端发送请求，onreadystatechange 属性的值为 goCallback。服务器端返回响应信息后，就会调用函数 goCallback()处理。goCallback()函数调用 setTimeout()函数每隔 2 秒的时间调用 pollServer()。

在 checkDiv()函数中，首先获取 ID 名称为 progressBar 的层对象。如果层已经显示，即“progress_bar.style.visibility == "visible"”，则调用函数 clearBar()清除进度条，并设置 ID 名称为 complete 的单元格值为空。如果该层不显示，则设置 visibility 属性值为 visible。

在 clearBar()函数中，主要完成使用 for 循环设置不同的 span 元素背景色为白色。在 for 循环内使用 document 对象的 getElementById()方法获取不同的 span 元素对象，并将获取的 span 元素对象使用空格代码，即 clear，最后设置这些 span 元素的背景色为白色。

（5）实现发送第二次请求和处理。客户端第一次发送请求后就会等待服务器端的响应信息，并调用处理响应信息的函数 goCallback()，goCallback()函数就会调用第二次发送请求的函数 pollServer()。实际上第一次请求并没有做任何实质性的工作，只是显示进度条并清空提示信息。通过 setTimeout()函数的使用，第二次请求实际上是多次循环发送的。第二次请求的任务是连续不断地发送数据，客户端依据服务器端的返回信息显示不同的 span 元素块。该步骤程序如代码 17.9 所示。

代码 17.9　显示不同的 span 元素块代码

```
         //刷新，重新发送
      function pollServer(){
            createXMLHttpRequest();
                  var url = "17.3-2.php?task=poll";
                  xmlHttp.open("GET",url,true);
                  xmlHttp.onreadystatechange = pollCallback;
                  xmlHttp.send(null);
            }
    //回调方法
          function pollCallback(){
            if (xmlHttp.readyState == 4){
             if (xmlHttp.status == 200){ //完成百分比
    percent complete = xmlHttp.responseXML.
    getElementsByTagName("percent")[0].firstChild.data;
                                   var index = processResult(percent_complete);
                                   for(var i=1;i<=index;i++){
                                    var elem = document.getElementById("block" + i);
```

```
                                        elem.innerHTML = clear;
                                        elem.style.backgroundColor = bar_color;
                                        var next_cell = i + 1;
                                        if ((next_cell > index)&&(next_cell<=9))
                                            {
    document.getElementById("block" + next_cell).innerHTML = percent_complete +
    "%";
                                            }
                                }
                                if (index < 9){
                                 setTimeout("pollServer()",2000);
                                }else{
                                 document.getElementById("complete").innerHTML =
                                 "完成";
                                 document.getElementById("go").disabled = false;
                                }
                        }
                }
        }
        function processResult(){//解析结果
                var ind;
                if(percent_complete.length == 1){
                   ind = 1;
                }else if (percent_complete.length == 2){
                   ind = percent_complete.substring(0,1);
                }else{
                   ind = 9;
                }
                return ind;
        }
```

在 pollServer()函数中同样调用 createXMLHttpRequest()创建异步传输对象，并创建变量 url。接下来使用 open()方法发送请求，并设置 pollCallback()为响应处理函数，清空。

在 pollCallback() 函数中首先使用 getElementsByTagName() 方法，获取响应信息 xmlHttp.responseXML 中名称为 percent 的节点集的第一个节点的子节点值，并将其赋值给 percent_complete 变量。接下来调用 processResult()函数将获取的 percent_complete 变量进行解析，并将获取的解析值赋值给变量 index，然后在 for 循环中使用 document 对象的 getElementById()方法获取不同的 span 元素对象。语句“elem.style.backgroundColor = bar_color”设置进度条运行时的颜色。创建变量 next_cell，其值为循环值 i 加 1。如果 next_cell 的值处于 1 到 9 之间，则在即将要显示颜色的 span 元素上显示进度百分比。如果 index 的值小于 9，则循环调用 pollServer()函数；否则在 ID 名称为 complete 的单元格上显示“完成”字符串信息。

在 processResult()函数中判断从服务器端返回数据的长度，如果长度为 1，则变量 ind 的值为 1；如果长度为 2，则调用 substring()函数获取返回信息的第一位数，如果值为 64 则获取 6；如果长度不是 1 和 2，则变量 ind 的值为 9。

（6）实现服务器端程序。在 17.3-2.php 文件中，需要根据客户端发送的多次请求返回信息。即从第一次请求开始进行简单的计数，并根据不同的计数，返回不同的响应信息。此操作实现的一个难点是页面的全局变量，因为 PHP 页面没有实际意义上的全局变量，故此处使用 session 会话注册了一个变量实现。实现服务器端程序如代码 17.10 所示。

代码 17.10　实现服务器端代码程序

```
<?php
session_start();
header('Content-type: text/xml;charset=GB2312');
header("cache-control:no-cache,must-revalidate");
if(!session_is_registered("counter")){
   session_register("counter");
   $_SESSION['counter']=1;
}
else{
  $counter=$_SESSION['counter'];
  $task=$_GET["task"];
  $res="";
  if($task=="create"){
     echo $task;
     $res="<key>1</key>";
     $counter=1;
   }else{
    $percent="";
     switch($counter){
        case 1:$percent = "10";break;
        case 2:$percent = "23";break;
        case 3:$percent = "35";break;
        case 4:$percent = "51";break;
        case 5:$percent = "64";break;
        case 6:$percent = "73";break;
        case 7:$percent = "89";break;
        case 8:$percent = "100";break;
      }
     $counter++;
     if($counter==9)
        $counter=1;
     $_SESSION['counter']=$counter;
     $res="<percent>".$percent."</percent>";
     echo "<?xml version='1.0' encoding='GB2312'?>";
     echo "<response>";
     echo $res;
     echo "</response>";
    }
```

```
    }
?>
```

在该文件中首先启动了一个页面会话，并使用 header()函数设置了页面响应类型、编码和是否缓存，然后判断页面中是否注册了一个 counter 会话变量。如果没有，则使用“session_register("counter")”语句注册一个会话变量，并使用“$_SESSION['counter']=1”设置 counter 变量的初始值为 1；如果已经注册，则获取注册的会话变量 counter 的值。最后使用$_GET[]获取 task 传递的值，如果其值为 create 则输出 task 的值，创建一个<task>节点，其值为 1。如果其值不为 create 则首先创建变量$percent，并从会话中获取的$coutner 的值。然后在多分支语句中查询和其匹配的选项，最后根据不同的$counter 值，给$percent 赋值。每执行一次多分支语句，$counter 的值就加 1。如果$coutner 的值增加到 9，则重新给$counter 赋值为 1，并将此变量值注册到会话中。最后将获取的值以 XML 文件格式输出，其值包含在节点<response>中。

（7）部署和运行。将上述两个文件保存到 F:\MyWeb\apache\htdocs\PHP\17.3 目录下，打开 IE 浏览器，在地址栏中输入“http://localhost/PHP/17.3/17.3-1.php”，单击【转到】按钮，会显示相应页面。单击该页面的【运行】按钮，显示效果如图 17-5 所示。



图 17-5　进度条显示

17.4　提供自动提示功能

许多人都使用 Google 网站，当在搜索提示框输入要查询的信息后，就会根据以前输入的搜索项显示信息。这就是使得数据输入更快、更容易，而且不容易出错。对于富客户端应用，增加这个功能可能很容易，但是传统 Web 应用长期以来都没有这个特性，Ajax 技术出现之后自动提示功能得到了实现。本节将创建一个带有自动提示功能的示例，该示例的具体开发步骤如下所示。

（1）打开记事本，创建 17.4-1.php 和 17.4-2.php 页，用来实现自动提示功能的客户端文件和服务器端文件。

（2）打开 17.4-1.php 文件，首先实现页面内容显示，该页面布局程序如代码 17.11 所示。

代码 17.11 页面布局代码

```
<fieldset><legend>Ajax 实现自动提示功能</legend>
   名称: <input type="text" size="20" id="names"
                        onkeyup="findNames();" style="height:20;" />
      <div style="position:absolute;" id="popup">
         <table id="name_table" bgcolor="#FFFAFA" border="0"
                        cellspacing="0" cellpadding="0"/>
         <tbody id="name_table_body"></tbody>
      </table>
   </div>
<br><br><br><br><br>
</fieldset>
```

上述代码创建了一个表单元素文本域 ID 名称为 names，并注册了键盘事件，即在文本域中输入信息就会调用 findNames()函数。接下来创建一个层、表格和单元格，其 ID 名称分别为 popup、name_table 和 name_table_body。

（3）实现 CSS 样式表。在执行数据查询时会根据前面的输入显示后面可能出现的信息，这些待选择的显示信息其样式需要区别于其他信息。其 CSS 样式信息代码如下所示：

```
<style type="text/css">
.mouseOut {
background: #708090;
color: #FFFAFA;
}
.mouseOver {
background: #FFFAFA;
color: #000000;
}
</style>
```

（4）实现异步传输对象的创建和发送。当触发键盘事件后就会将输入的信息作为参数发送给服务器端程序，并进入等待服务器端响应状态。在发送信息之前需要创建异步传输对象，如代码 17.12 所示。

代码 17.12 实现异步传输程序

```
<script type="text/javascript">
   var xmlHttp;
   var completeDiv;
   var inputField;
   var nameTable;
   var nameTableBody;
   function createXMLHttpRequest() {//创建异步传输对象
         if (window.ActiveXObject) {
      xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
```

```
        }
        else if (window.XMLHttpRequest) {
            xmlHttp = new XMLHttpRequest();
        }
    }
    function initVars() {//初始化成员变量
        inputField = document.getElementById("names");
        nameTable = document.getElementById("name_table");
        completeDiv = document.getElementById("popup");
        nameTableBody = document.getElementById("name_table_body");
    }
    function findNames() {//发送请求
        initVars();
        if (inputField.value.length > 0) {
            createXMLHttpRequest();
            var url = "17.4-2.php?names=" + escape(inputField.value);
            xmlHttp.open("GET", url, true);
            xmlHttp.onreadystatechange = callback;
            xmlHttp.send(null);
        } else {
            clearNames();
        }
    }
    function clearNames() {//清除信息
        var ind = nameTableBody.childNodes.length;
        for (var i = ind - 1; i >= 0 ; i--) {
             nameTableBody.removeChild(nameTableBody.childNodes[i]);
        }
        completeDiv.style.border = "none";
    }
    </script>
```

在 findNames()函数中，首先调用 initVars()函数初始化各个成员变量的值，即使用 document.getElementById 方法分别获取文本域、层、表格和单元格对象。接下来进行判断，如果文本域中值的长度大于 0，调用 createXMLHttpRequest()函数创建异步传输对象，并创建 url，其值为服务器端程序和文本域参数。然后使用 GET 方法向服务器端发送请求，设定 onreadystatechange 属性的值为 callback，即设定处理服务器端响应的函数。send()方法表示向服务器端发送空信息。如果文本域中值的长度小于等于 0，则调用 clearNames()函数。

在 clearNames()函数中，首先获取单元格对象 nameTableBody 的子节点的长度，并以子节点的长度作为循环条件。在 for 循环内依次使用 nameTableBody 对象的 removeChild()方法移除单元格中的子节点，即上一次显示的信息。最后使用 completeDiv.style.border 设定层的边框为无。

（5）实现服务器端响应信息的处理。客户端获取了服务器端响应信息之后就会调用 setNames()方法和相关函数处理响应信息，如代码 17.13 所示。

代码 17.13　setNames()方法和相关函数处理响应信息程序

```
    function callback() {
        if (xmlHttp.readyState == 4) {
            if (xmlHttp.status == 200) {
                var name = xmlHttp.responseXML.getElementsByTagName("name");
                setNames(name);
            } else if (xmlHttp.status == 204){
                clearNames();
            }
        }
    }
function setNames(the_names) {
        clearNames();
        var size = the_names.length;
        setOffsets();
        var row, cell, txtNode;
        for (var i = 0; i < size&& i<10; i++) {
        //如果超过 10 条记录，只显示前 10 条
            var nextNode = the names[i].firstChild.data;
            row = document.createElement("tr");
            cell = document.createElement("td");
            cell.onmouseout = function() {this.className='mouseOut';};
            cell.onmouseover = function() {this.className='mouseOver';};
            cell.setAttribute("bgcolor", "#FFFAFA");
            cell.setAttribute("border", "0");
            cell.onclick = function() { populateName(this); } ;
            txtNode = document.createTextNode(nextNode);
            cell.appendChild(txtNode);
            row.appendChild(cell);
            nameTableBody.appendChild(row);
        }
    }
    function setOffsets() {
        var end = inputField.offsetWidth;
        var left = calculateOffsetLeft(inputField);
        var top = calculateOffsetTop(inputField) + inputField.offsetHeight;
        completeDiv.style.border = "black 1px solid";
        completeDiv.style.left = left + "px";
        completeDiv.style.top = top + "px";
        nameTable.style.width = end + "px";
    }
    function calculateOffsetLeft(field) {
      return calculateOffset(field, "offsetLeft");
    }
```

```
        function calculateOffsetTop(field) {
          return calculateOffset(field, "offsetTop");
        }
        function calculateOffset(field, attr) {
          var offset = 0;
          while(field) {
            offset += field[attr];
            field = field.offsetParent;
          }
          return offset;
        }
        function populateName(cell) {
            inputField.value = cell.firstChild.nodeValue;
            clearNames();
        }
```

函数 callback()首先从服务器端的响应信息 xmlHttp.responseXML 中获取节点名称为 name 的节点集合对象，然后将 name 作为参数调用 setName()函数。如果服务器端出现错误，则调用 clearNames()函数。

在 setName()函数中，首先调用 clearNames()函数清除上一次显示信息，并创建 size 获取参数 the_names 传递的响应信息长度。接下来调用 setOffsets()函数设定要显示信息的显示样式。在 for 循环内，使用语句“the_names[i].firstChild.data”获取服务端的响应信息；创建两个变量 row 和 cell，分别用来表示新创建的 HTML 节点<tr>和<td>，然后使用 cell 对象设置提示数据的显示信息样式，例如采用的选择器。使用 setAttribute()方法设置背景色和边框等。创建 txtNode 变量用来表示新创建的文本节点，并将此文本节点追加到 cell 单元格上，cell 单元格追加到<row>节点上，最后将<row>节点追加到 nameTableBody 单元格上。

setOffsets()、calculateOffsetLeft(field)、calculateOffsetTop(field)和 calculateOffset(field, attr)函数主要用来设定提示数据显示样式的大小，这里就不再介绍了。populateName(cell)主要将获取的服务器端响应信息的第一个子节点赋值给文本域。

（6）实现服务器端程序。服务器端程序需要根据客户端提交的信息，返回不同的响应信息，如输入“a”则需要返回 ajax，abc 和 apple 等。服务器端程序如代码 17.14 所示。

代码 17.14　实现服务器端程序

```
<?php
header('Content-Type: text/xml');
header("cache-control:no-cache,must-revalidate");
$xml="<?xml version='1.0' encoding='GB2312'?>";
$name=$ GET["names"];
if($name=="a"){
$data="<response><name>abc</name><name>ajax</name><name>apple</name><name>
an</name></response>";
   echo $xml.$data;
```

```
      }
    if($name=="Y")
      {
    $data="<response><name>Yound</name><name>Yound Tang</name>
    <name>You</name><name>Yang</name></response>";
       echo $xml.$data;
      }
    ?>
```

在上述代码中，使用 header()函数设定页面响应信息格式为 text/xml 格式，以及不允许页面缓存。接下来创建变量$xml，其值为 XML 文件头。然后使用$_GET 获取客户端传递的 names 参数值，如果参数值为 a 则创建变量$data，其值由<response>节点和子节点<name>组成，并将$xml 和$data 字符串连接并输出。如果参数值为 Y，同样也是创建变量$data，并输出。

（7）部署和运行。将上述两个文件保存到 F:\MyWeb\apache\htdocs\PHP\17.4 目录下。打开 IE 浏览器，在地址栏中输入“http://localhost/PHP/17.4/17.4-1.php”，单击【转到】按钮，会显示相应页面。在页面的文本域中输入“Y”，会显示与 Y 相关的提示信息，此时页面效果如图 17-6 所示。

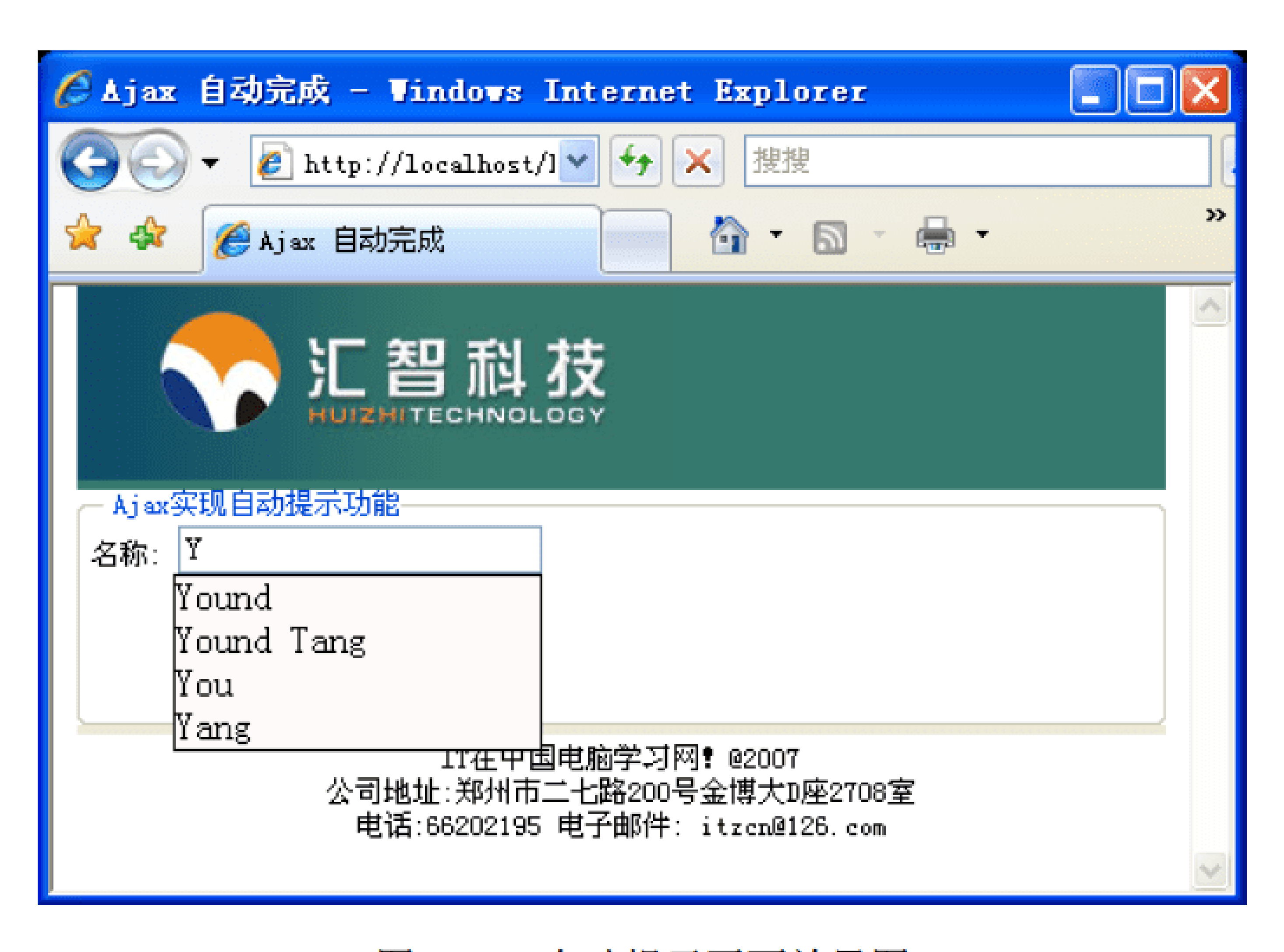


图 17-6　自动提示页面效果图

17.5　完成数据库各项操作

经常需要将一些资料信息发布到网上，以便用户及时查看最新信息。这时就需要使用 PHP 这些后台语言来实现快捷更新、删除和增加信息，以及管理数据库相关信息。本节将创建一个示例具体讲解如何进行这些操作，具体步骤如下所示。

（1）实现数据库和数据表。首先在 MySQL 服务器上创建一个 books 数据库，然后在该数据库中创建一个 mybook 表，该数据库表字段介绍如表 17-1 所示。

表 17-1 表 mybook 字段介绍

字段	数据类型	是否为空	备注
id	tinyint unsigned	否	图书 ID
name	Varchar(45)	否	图书名称
author	Varchar(45)	否	图书作者
publisher	Varchar(45)	否	图书出版社
isbn	Varchar(45)	否	图书 ISBN 号
type	Varchar(45)	否	图书类型
price	tinyint	否	图书价格

（2）打开记事本，创建 chakan.php 和 getChakan.php，用来实现数据显示操作。

（3）打开 chakan.php 文件，首先实现 PHP 页面内容显示和操作数据链接。该步骤程序如代码 17.15 所示。

代码 17.15 数据显示和操作链接

```
<body onLoad="validate()" style="font-family: '新宋体'; font-size: 12px;
margin:0;">
<center><div style="width:900px;"><fieldset><legend>图书显示</legend>
<div id="res"></div>
<p align="center">|<a href="addBook.php">数据添加</a>|<a href="deleteBook.
php">数据删除</a>|<a href="update.php">数据修改</a>|</p> </fieldset>
</div></center></body>
```

上述代码创建了几个超级链接，用来完成指定操作。接下来创建了一个 ID 名称为 res 的层。页面打开时会加载 validate()函数。

（4）实现数据显示的请求发送和响应处理。如果要在 chakan.php 页面显示数据库当前信息，客户端需要向数据库发送一个显示请求，并根据服务器的响应信息设定数据的显示样式。该步骤程序如代码 17.16 所示。

代码 17.16 显示数据处理代码

```
<?php
header("cache-control:no-cache,must-revalidate");
?>
<html>
<head>
<script type="text/javascript">
var xmlHttp;
function createXMLHttpRequest(){
   if(window.ActiveXObject){
     xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
   else if(window.XMLHttpRequest){
      xmlHttp=new XMLHttpRequest();
   }
```

```
    }
    function validate(){
        createXMLHttpRequest();
        var url="getChakan.php";
        xmlHttp.open("GET",url,true);
        xmlHttp.onreadystatechange=callback;
        xmlHttp.send(null);
    }
    function callback(){
        if(xmlHttp.readyState==4){
            if(xmlHttp.status==200){
             show();
            }
        }
    }
    function show(){
        //alert(xmlHttp.responseXML.xml);

        var xmlDoc=xmlHttp.responseXML;
        var yan=xmlDoc.getElementsByTagName("content");
        var ta="<table style='width:100%;background-color: #F9FBFD;
        border-top-width: 1px;border-top-style: solid;  border-right-style:
        solid;border-bottom-style: solid;    border-left-style: solid;
        border-top-color: #CDD9E7;   border-right-color: #CDD9E7;
        border-bottom-color: #CDD9E7;    border-left-color: #CDD9E7;
        border-right-width: 1px; border-bottom-width: 0px; border-left-width:
        1px;'><th>名称</th><th>作者</th><th>出版社</th><th>ISBN 号</th><th>图书类型
        </th><th>价格</th>";
        for(var i=0;i<yan.length;i++){
            var y=yan[i];
            ta+="<tr><td>"+y.childNodes[1].firstChild.data+"</td>";
            ta+="<td>"+y.childNodes[2].firstChild.data+"</td>";
            ta+="<td>"+y.childNodes[3].firstChild.data+"</td>";
            ta+="<td>"+y.childNodes[4].firstChild.data+"</td>";
            ta+="<td>"+y.childNodes[5].firstChild.data+"</td>";
            ta+="<td>"+y.childNodes[6].firstChild.data+"</td></tr>";
        }
        ta+="</table>";
        document.getElementById("res").innerHTML=ta;


    }
    </script></head></html>
```

在 validate()函数中，首先调用 createXMLHttpRequest()函数实现异步传输对象的创建，并

指定显示操作的服务器程序 getChakan.php，即创建变量 url。然后设定处理服务器端响应信息的函数为 callback()，并向服务器发送一个请求。

在 callback()函数中，调用 show()函数处理响应信息。在 show()函数中，使用 xmlHttp.responseXML 获取服务器端返回的 XML 文件对象 xmlDoc，并使用 getElementsByTagName()方法获取节点名称为 content 的节点集对象 yan。接下来创建变量 ta，用来设置数据显示的表格演示。在 for 循环中以节点集 yan 的长度为退出条件，yan[i]表示节点集中某一个 content 节点。“y.childNodes[1].firstChild.data”语句表示获取<content>节点的第二个子节点的第一个子节点的值，其中 childNodes[1]表示 content 的第二个子节点，依次类推，childNodes[0]表示 conent 的第一个子节点，y.childNodes[2]表示获取 content 的第 3 个子节点。最后将获取的 ta 字符串赋值给名称为 res 的层。

（5）实现服务器端程序。客户端发送请求之后，服务器端程序会将 mybook 数据表提取处理，并以 XML 文件返回到客户端。该步骤程序如代码 17.17 所示。

代码 17.17　服务器端代码

```
<?php
 header('Content-Type: text/xml');
 header("cache-control:no-cache,must-revalidate");
  $xml="<?xml version='1.0' encoding='GB2312'?>";
  $xml=$xml."<contents>";
  $link=mysql_connect("localhost","root","0");
  mysql_select_db("books");
  $exec="select * from mybook";
  $result=mysql_query($exec);
 while($rs=mysql_fetch_object($result))
     {
       $id=$rs->id;
       $name=$rs->name;
       $author=$rs->author;
       $pub=$rs->publisher;
       $isbn=$rs->isbn;
       $type=$rs->type;
       $price=$rs->price;
     $content=$content."<content><id>".$id."</id><name>".$name."</name>
     <author>".$author."</author><publisher>".$pub."</publisher><isbn>".
     $isbn."</isbn><type>".$type."</type><price>".$price."</price>
     </content>";
  }

$xml=$xml.$content."</contents>";
echo $xml;
   mysql_close();
?>
```

在上述代码中，首先使用 header()函数设置页面返回类型和不允许缓存。创建变量$xml，其值为 XML 文件头，并为$xml 追加了一个<contents>字符串。接下来代码主要将数据从 mybook 中读取出来。在 while 循环中将获取的字段值分别赋值给不同的变量，然后将这些变量追加到字符串$content 中，最后以 XML 文件形式输出。

（6）数据显示操作的部署和运行。将上述两个文件保存到 F:\MyWeb\apache\htdocs\PHP\17.5\chakan 目录下，打开 IE 浏览器，在地址栏中输入“http://localhost/PHP/17.5/chakan.php”，单击【转到】按钮，页面显示效果如图 17-7 所示。



图 17-7　图书信息显示

（7）打开记事本，创建 addBook.php 和 getAddBook.php 文件，用来实现数据添加操作。

（8）打开 addBook.php 文件，该页面首先实现了用户输入信息的表单。由于布局比较简单，在此就不再提供代码。

（9）实现 addBook.php 页面插入请求和响应处理。在客户端输入数据之后就可以单击【添加】按钮提交并发送数据了，在发送数据之前需要进行简单信息校验。如果信息校验通过则发送信息，否则重新输入。客户端依据服务器端的响应显示不同的信息。该步骤程序如代码 17.18 所示。

代码 17.18　客户端代码

```
<script type="text/javascript">
function validate form()
{
  if(document.form1.name.value == "")
    {
        alert("图书名称不能为空!");
        return false;
    }
  if(document.form1.author.value == "")
    {
        alert("图书作者不能为空!");
        return false;
```

```
    }
  if(document.form1.isbn.value == "")
    {
        alert("ISBN 号不能为空!");
        return false;
    }
  if(document.form1.price.value == "")
    {
        alert("图书价格不能为空");
        return false;
    }
      return true;
}
function tian()
{
    if(validate form()==false)
        return;
    validate();
}
var xmlHttp;
function createXMLHttpRequest(){
  if(window.ActiveXObject){
    xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  else if(window.XMLHttpRequest){
     xmlHttp=new XMLHttpRequest();
  }
}
function validate(){
  createXMLHttpRequest();
  var name=document.getElementById("name").value;
  var author=document.getElementById("author").value;
  var pl=document.getElementById("publisher");
  var publisher=pl.options[pl.selectedIndex].text;
  var isbn=document.getElementById("isbn").value;
  var tp=document.getElementById("type");
  var type=tp.options[tp.selectedIndex].text;
  var price=document.getElementById("price").value;
 var str="name="+name+"&author="+author+"&publisher="+publisher+"&isbn=
 "+isbn+"&type="+type+"&price="+price;
  var url="getAddBook.php";
  xmlHttp.open("POST",url,true);
  xmlHttp.onreadystatechange=callback;
  xmlHttp.setRequestHeader("Content-Type","application/x-www-form-
  urlencoded");
```

```
       xmlHttp.send(str);
    }

    function callback(){
       if(xmlHttp.readyState==4){
           if(xmlHttp.status==200){
               var str=xmlHttp.responseText;
               if(str=="1"){
                     alert("图书数据已经添加");
                        }
                else {
                       alert("添加失败");
                        }
               window.location.reload();

        }}
    }
    </script>
```

在函数 tian()中，首先调用 validate_form()函数判断用户输入数据是否符合要求，例如非空验证。如果不符合要求则返回 false，不做任何操作；如果符合要求则调用 validate()函数发送请求。在 validate()函数中，使用“getElementById("XXX").value”方式获取表单中的输入值，其中 XXX 为表单元素的 ID 名称。如果要获取下拉列表的值，需要下拉列表的对象，如“document.getElementById("type")”语句，再使用“tp.options[tp.selectedIndex].text”语句获取被选中项的值。然后将获取的所有输入信息组合成一个字符串 str，并创建变量 url，其值为服务器处理程序。最后向服务器程序发送请求，设定处理服务器端响应的函数为 callback()，并使用 send()函数将字符串 str 发送到服务器端。在 callback()函数中，首先使用 xmlHttp.responseText 获取服务器端的响应信息，如果获取的信息值为 1 则弹出提示信息，并重新加载本页面。

（10）实现数据添加操作的服务器端程序。服务器端获取了客户端提交的信息，需要将信息进行中文转换，然后将数据插入到数据表 mybook 中。该步骤程序如代码 17.19 所示。

代码 17.19　实现插入数据库服务器端代码

```
    <?php
    header('Content-type: text/html;chareset:GB2312');
    $name=$_POST['name'];
    $author=$_POST['author'];
    $publisher=$_POST['publisher'];
    $isbn=$_POST['isbn'];
    $type=$_POST['type'];
    $price=$_POST['price'];
    $name=iconv( 'UTF-8', 'GB2312       //IGNORE' , $name);
    $author=iconv( 'UTF-8', 'GB2312     //IGNORE' , $author);
```

```
$publisher=iconv( 'UTF-8', 'GB2312 //IGNORE' , $publisher);
$isbn=iconv( 'UTF-8', 'GB2312       //IGNORE' , $isbn);
$type=iconv( 'UTF-8', 'GB2312       //IGNORE' , $type);
$price=iconv( 'UTF-8', 'GB2312      //IGNORE' , $price);
$link=mysql_connect("localhost","root","0");
mysql_select_db("books");
$exec="insert into mybook (name,author,publisher,isbn,type,price) values
('".$name."','".$author."','".$publisher."','".$isbn."','".$type."','".
$price."')";
$result=mysql_query($exec);
if($result){
        echo "1";
    }
  mysql_close();
?>
```

在代码 17.19 中，首先使用 header()函数设定页面编码为 GB2312，并使用$_POST 获取客户端传递的参数值，如 name 和 author 等。为了保证不出现中文乱码，还需要使用 iconv()函数对数据进行处理，“iconv('UTF-8', 'GB2312//IGNORE' , $name)”语句表示将$name 字符串 UTF-8 的编码格式转换为 GB2312 的编码格式。其中 IGNORE 的意思是忽略转换时的错误，如果没有 IGNORE 参数，所有该字符后面的字符串都无法被保存，即会出现字符丢失的现象。然后调用“mysql_query("set names 'GB2312'")”，设定数据编码为 GB2312。接下来的数据库添加操作前面已经介绍过，这里就不再介绍了。如果数据添加成功，就会输出字符串“1”。

（11）添加操作的部署和运行。单击图 17-7 中的【数据添加】超级链接，输入信息后单击【添加】按钮，添加成功后页面显示效果如图 17-8 所示。

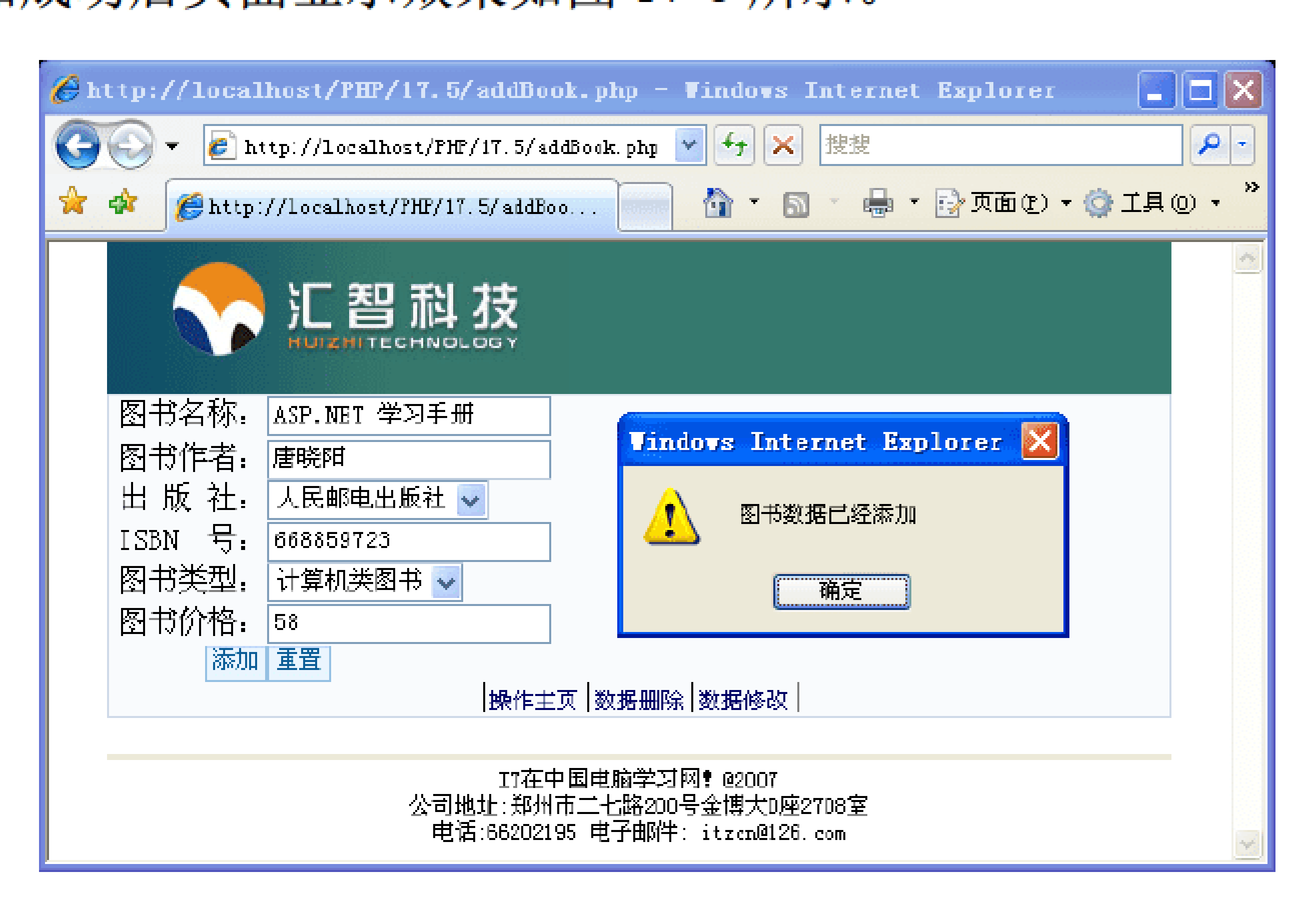


图 17-8　数据添加显示页面

（12）打开记事本，创建 deleteBook.php 和 getDeleteBook.php 文件，用来实现图书信息删除操作的客户端程序和服务器端程序。

（13）打开 deleteBook.php 文件，首先实现页面内容显示，由于该页面布局比较简单，在此只提供绑定数据的 PHP 程序代码。该步骤主要程序如代码 17.20 所示。

代码 17.20　PHP 获取数据代码

```
<?php
 $link=mysql_connect("localhost","root","0");
  mysql_select_db("books");
  $exec="select * from mybook";
  $result=mysql_query($exec);
 while($rs=mysql_fetch_object($result))
     {
       $id=$rs->id;
       $name=$rs->name;
       $author=$rs->author;
       $pub=$rs->publisher;
       $isbn=$rs->isbn;
       $type=$rs->type;
       $price=$rs->price;
     $content="<tr><td>".$name."</td><td>".$author."</td><td>".$pub.
     "</td><td>".$isbn."</td><td>".$type."</td><td>".$price."</td><td>
     <p id='".$id."' onclick='Start(this)' style='color:blue;cursor:hand'>
     删除</p></td></tr>";
echo $content;
  }
?>
```

上述代码主要完成将 mybook 表中的数据提取出来，并在表格中显示。代码“<p id='".$id."' onclick='Start(this)' style='color:blue;cursor:hand'>删除</p>”表示创建一个 HTML 段落，其 ID 名称为变量$id 的值，并给该段落注册了一个 onclick 事件，即当单击该段落时会调用函数 Start(this)，this 指向当前对象；style 表示给段落设置一个样式。

（14）实现发送请求和响应信息处理。在 deleteBook.php 文件中将 mybook 表中的数据显示出来后，就可以删除指定的选项了。当单击【删除】文本时，就会调用 Start()函数向服务器发送一个请求，并处理服务器端响应信息。该步骤脚本程序如代码 17.21 所示。

代码 17.21　脚本处理程序

```
<script type=text/javascript>
var xmlHttp;
function createXMLHttpRequest(){
   if(window.ActiveXObject){
     xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
   else if(window.XMLHttpRequest){
      xmlHttp=new XMLHttpRequest();
   }
}
function Start(ele){
```

```
    id=ele.id;
    createXMLHttpRequest();
    var url="getDeleteBook.php?key="+escape(ele.id);
    xmlHttp.open("GET",url,true);
    xmlHttp.onreadystatechange=Del;
    xmlHttp.send(null);
  }
  function Del(){
     if(xmlHttp.readyState==4){
           if(xmlHttp.status==200){
             if(xmlHttp.responseText=="1")
            {
              alert("该项已被删除");
              window.location.reload();
            }
             else{
                 alert("页面可能出现错误，请重新试一下");
               }
         }
       }
  }
  </script>
```

在 Start()函数中，首先创建变量 id 获取单击时传递的参数，即要删除的图书信息的 ID 号，其中 ele 表示传递的段落对象，ele.id 表示该对象的 id 名称。接下来调用 createXMLHttpRequest()函数创建异步传输对象，并创建变量 url，其值为服务器端程序以及传递的参数值。最后使用 open()方法发送请求，设定处理服务器端响应的函数为 Del()，并向服务器端发送空信息。在函数 Del()中，如果返回的响应信息为 1 则提示信息被删除，并重新加载该页面，否则提示错误信息。

（15）实现删除操作的服务器端程序。在 getDeleteBook.php 文件中需要根据客户端传递的参数，创建相应的删除语句，执行数据记录的删除。该步骤程序如代码 17.22 所示。

代码 17.22　服务器端删除数据代码

```
<?php
$id=$_GET['key'];
$link=mysql_connect("localhost","root","0");
mysql_select_db("books");
$exec="delete from mybook where id=$id";
$result=mysql_query($exec);
if((mysql_affected_rows()==0) || (mysql_affected_rows()==-1))
    {
            echo "0";
            exit;
       }
```

```
    else{
        echo "1";
        }
        mysql_close();
?>
```

（16）删除操作的部署和运行。单击图 17-7 中的【数据删除】超级链接，会显示如图 17-9 所示。在显示的页面上选择要删除的数据库记录，单击相应的【删除】超级链接，如果成功会显示删除成功对话框。

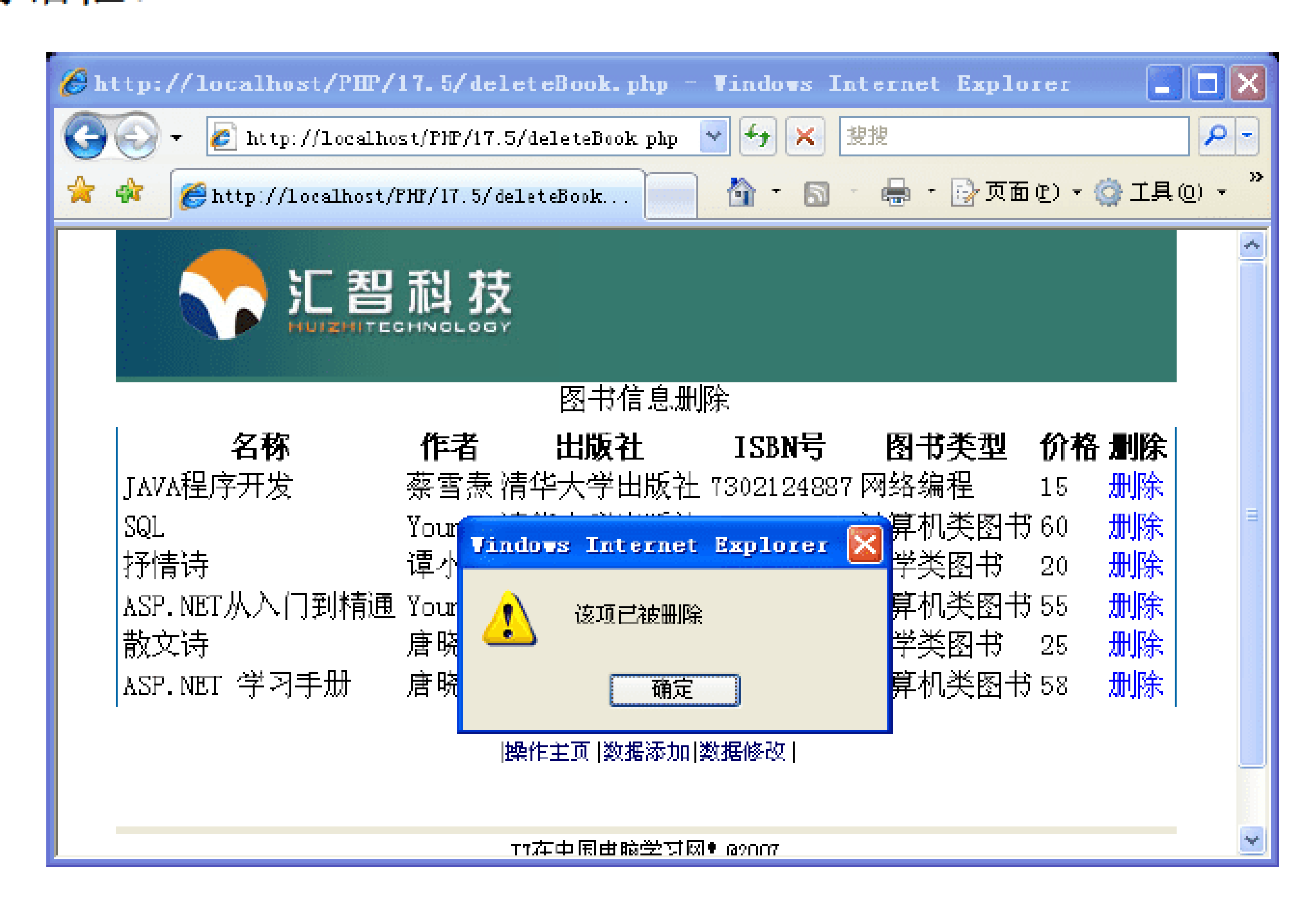


图 17-9　删除信息页面

（17）打开记事本，创建 update.php、UpdateBook.php 和 GetUpdateBook.php 文件，用来实现选择修改记录、修改记录显示页面和执行修改操作等。

（18）实现选择修改记录操作。在 update.php 文件中需要实现显示当前所有的图书信息，并将其列到当前页面上。其绑定数据的 PHP 程序如代码 17.23 所示。

代码 17.23　显示数据代码

```
<table border=1 width=68% align=center>
<caption>图书信息修改</caption>
<th>名称</th><th>作者</th><th>修改</th>
<?php
 $link=mysql_connect("localhost","root","0");
  //mysql query("set names 'gbk'");
  mysql select db("books");
  $exec="select * from mybook";
  $result=mysql query($exec);
 while($rs=mysql_fetch_object($result))
     {
       $id=$rs->id;
       $name=$rs->name;
       $author=$rs->author;
```

```
        $content="<tr><td>".$name."</td><td>".$author."</td><td><a
        href='UpdateBook.php?id=".$id."'>修改</a></td></tr>";
        echo $content;
   }
?>
</table>
```

上述代码主要将 mybook 数据表中的信息提取，并将信息显示到当前页面的上面。其中代码“修改”表示创建一个超级链接，并传递一个参数 id，其为变量$id 的值。

（19）实现修改记录的显示。单击上个文件的【修改】超级链接，页面会转向 UpdateBook.php。UpdateBook.php 文件会根据传递的 id 值获取指定数据库记录的详细信息。该步骤主要程序如代码 17.24 所示。

代码 17.24　修改数据页程序

```
<?php
$id=$ GET['id'];
 $link=mysql connect("localhost","root","0");
  mysql_select_db("books");
  $exec="select * from mybook where id='".$id."'";
  $result=mysql_query($exec);
  while($rs=mysql_fetch_object($result))
     {
       $id=$rs->id;
       $name=$rs->name;
       $author=$rs->author;
       $pub=$rs->publisher;
       $isbn=$rs->isbn;
       $type=$rs->type;
       $price=$rs->price;
?>
<fieldset><legend>信息修改页面</legend></fieldset>
<form name="form1">
<table align="left">
<tr><td>图书名称：</td><td><input type=hidden id="id" value=<?php echo $id?>>
<input type="text" id="name" value=<?php echo $name?>></td></tr>
<tr><td>图书作者：</td><td><input type="text" id="author" value=<?php echo
$author?>></td></tr>
<tr><td>出 版 社：</td><td>
<select id=publisher>
<option><?php echo $pub?></option>
<option>清华大学出版社</option>
<option>人民邮电出版社</option>
<option>电子工业出版社</option>
```

```
<option>河南出版社</option>
</select>
</td></tr>
<tr><td>ISBN  号: </td><td><input type="text" id="isbn" value=
<?php echo $isbn?>></td></tr>
<tr><td>图书类型: </td><td>
<select id="type">
<option><?php echo $type?></option>
<option>计算机类图书</option>
<option>经济类图书</option>
<option>财经类图书</option>
<option>哲学类图书</option>
</select>
</td></tr>
<tr><td>图书价格: </td><td><input type="text" id="price" value=<?php echo
$price?>></td></tr>
<tr><td><input type="button" value="修改" onclick="xiu()"></td><td><input
type="reset" value="重置"></td></tr></table></form>
```

在上述代码中，首先使用$_GET 获取传递参数 id 的值，并根据该 id 值创建相应的查询语句，获取该记录的每个字段值。接下来将获取的字段值嵌入到表单元素中，如“<input type=hidden id="id" value=<?php echo $id?>>”表示将图书 id 的值作为隐藏按钮的值处理。同样也可以将获取的值嵌入到文本域和下拉列表中等，这样就可以在相应的表单元素中修改信息。“<input type="button" value="修改" onclick="xiu()">”表示单击【修改】按钮时会调用 JavaScript 函数 xiu()。

（20）实现修改信息的发送和响应处理。在 UpdateBook.php 中，文本域或下拉列表中修改完信息后，就可以单击【修改】按钮调用 xiu()函数发送请求，并调用 callback()函数处理响应。该步骤脚本程序如代码 17.25 所示。

代码 17.25　脚本程序

```
<script type="text/javascript">
var xmlHttp;
function createXMLHttpRequest(){
   if(window.ActiveXObject){
     xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
   else if(window.XMLHttpRequest){
      xmlHttp=new XMLHttpRequest();
   }
}

function xiu(){
   createXMLHttpRequest();
```

```
        var id=document.getElementById("id").value;
        var name=document.getElementById("name").value;
        var author=document.getElementById("author").value;
        var pl=document.getElementById("publisher");
        var publisher=pl.options[pl.selectedIndex].text;
        var isbn=document.getElementById("isbn").value;
        var tp=document.getElementById("type");
        var type=tp.options[tp.selectedIndex].text;
        var price=document.getElementById("price").value;
       var str="id="+id+"&name="+name+"&author="+author+"&publisher="+publisher+
       "&isbn="+isbn+"&type="+type+"&price="+price;
        var url="GetUpdateBook.php";
        xmlHttp.open("POST",url,true);
        xmlHttp.onreadystatechange=callback;
        xmlHttp.setRequestHeader("Content-Type","application/x-www-form-
        urlencoded");
        xmlHttp.send(str);
    }

    function callback(){
        if(xmlHttp.readyState==4){
            if(xmlHttp.status==200){
                var str=xmlHttp.responseText;
                if(str=="1")
                      alert("图书信息修改完毕");

     }}
    }



    </script>
```

在 xiu()函数中需要获取页面提交的修改信息，并发送给服务器端程序 GetUpdateBook.php 处理。其实现代码和步骤（9）中的代码基本相同，这里就不再赘述。

（21）实现修改信息操作的服务器端程序。服务器端程序需要依据提交的信息创建修改的 SQL 语句并执行。该步骤服务器端程序如代码 17.26 所示。

代码 17.26　服务器端程序代码

```
<?php
header('Content-type: text/html;chareset:GB2312');
$id=$_POST['id'];
$name=$_POST['name'];
```

```
$author=$_POST['author'];
$publisher=$ POST['publisher'];
$isbn=$_POST['isbn'];
$type=$_POST['type'];
$price=$_POST['price'];
$name=iconv( 'UTF-8', 'GB2312      //IGNORE' , $name);
$author=iconv( 'UTF-8', 'GB2312     //IGNORE' , $author);
$publisher=iconv( 'UTF-8', 'GB2312 //IGNORE' , $publisher);
$isbn=iconv( 'UTF-8', 'GB2312      //IGNORE' , $isbn);
$type=iconv( 'UTF-8', 'GB2312      //IGNORE' , $type);
$price=iconv( 'UTF-8', 'GB2312     //IGNORE' , $price);
$link=mysql_connect("localhost","root","0");
//mysql_query("set names 'GB2312'");
mysql_select_db("books");
$exec="update mybook set name='$name',author='$author',publisher=
'$publisher',isbn='$isbn',type='$type',price='$price' where id='$id'";
$result=mysql_query($exec);
if($result){
        echo "1";
    }
  mysql_close();
?>
```

（22）修改操作的部署和运行。单击图 17-7 中的【数据修改】超级链接，会显示如图 17-10 所示。



图 17-10　图书修改操作页面

单击图 17-10 中记录项中的【修改】超级链接，记录项会显示到修改信息页面，如图 17-11 所示。如果修改成功，会显示提示信息对话框。

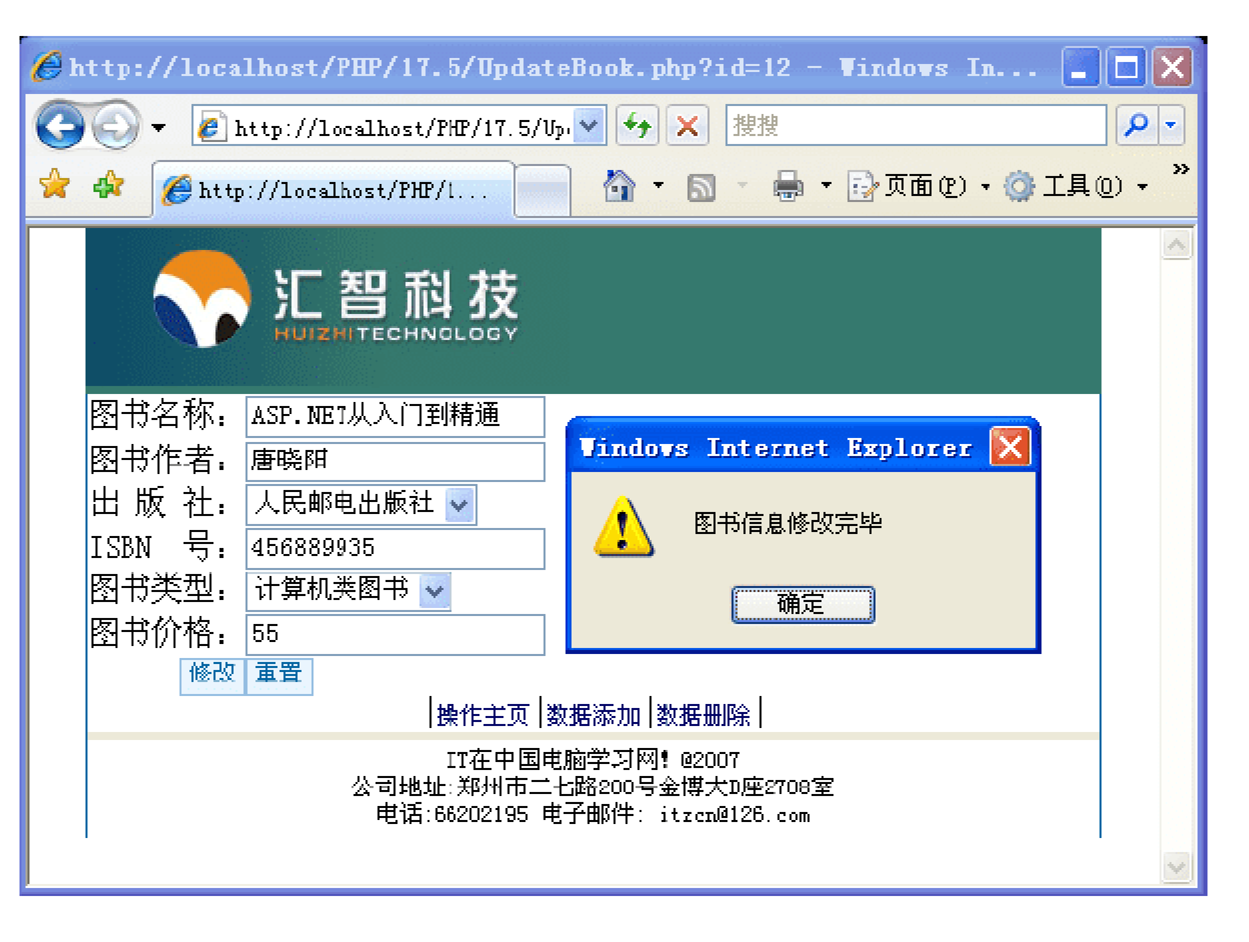


图 17-11　图书修改操作显示

17.6　级联菜单形式显示信息

级联下拉菜单就是从一个下拉菜单中选中一项后，另一个下拉菜单的内容会相应的随之改变。一般来说，最简单的就是每次选中都提交一次表单刷新整个页面。另一种是将所有选项在第一次加载时就全部载入整个页面中的 JavaScript 数组中，然后级联通过 JavaScript 来控制。在整个数据量不大时，这是一个不错的实现无刷新并且快速的方法，但是当整个数据量非常大时，这种方法就会使第一次加载变得非常慢。而采用 Ajax 方式，即开始只载入第一层菜单的内容，当用户选中第一层菜单的某项时，再通过 XMLHttpRequest 来获取相应选项所对应的第二层菜单的内容，这种效果最好。下面创建一个这样的示例，该示例具体步骤如下所示。

（1）打开记事本，创建 DemoSelect.php 和 DemoSelect1.php 文件，用来实现级联菜单的客户端文件和服务器端文件。

（2）打开 DemoSelect.php 文件，首先实现 PHP 页面内容，其代码如下所示：

```
<h1>分级下拉菜单示例</h1>
选择第一项：<select id="number" onchange="validate();">
           <option value="1">优秀学生</option>
           <option value="5">优秀团员</option>
        </select>
  <select id="dataMessage">
     <option>查看此处</option>
  </select>
```

上述代码创建了两个下拉列表，其 ID 名称分别为 number 和 dataMessage，并给第一个下拉列表注册了 onchange 事件，即当选择其中一项时就调用 JavaScript 函数 validate()。

（3）实现异步传输对象的创建和请求发送。当选择第一个下拉菜单的选项时就会触发下拉列表的 onchange 事件，即调用相关函数向服务器端发送一个请求。在调用请求之前，需要

创建异步传输对象。该步骤程序如代码 17.27 所示。

代码 17.27　创建异步传输对象

```
<script type="text/javascript">
var xmlHttp;
var a=new Array();
function createXMLHttpRequest(){
   if(window.ActiveXObject){
     xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
   else if(window.XMLHttpRequest){
      xmlHttp=new XMLHttpRequest();
   }
}
function validate(){
   createXMLHttpRequest();
   var date=document.getElementById("number");
   var url="Example7_7_1.php?birthDate="+escape(date.value);
   xmlHttp.open("GET",url,true);
   xmlHttp.onreadystatechange=callback;
   xmlHttp.send(null);
}
</script>
```

在 validate()函数中，首先调用 createXMLHttpRequest()函数创建异步传输对象，并创建变量 date 获取第一个下拉列表框中被选中的选项。接下来创建 url 变量，用来设定发送请求的目的程序和参数，使用 open()方法发送请求。最后设定 onreadystatechange 属性的值 callback。

（4）实现响应信息的处理。当客户端获取了服务器端的响应信息后，就调用 callback()以及其他函数处理响应信息。该步骤程序如代码 17.28 所示。

代码 17.28　实现级联菜单绑定

```
function callback(){
   if(xmlHttp.readyState==4){
     show();
   }
}
function show(){
   var xmlDoc=xmlHttp.responseXML;
   var content=xmlDoc.getElementsByTagName("student");
   for(var i=0;i<content.length;i++){
      var y=content[i];
      a[i]=y.childNodes[0].data;
   }
  show1();
```

```
    }
    function show1(){
        var obj=document.getElementById("dataMessage");
        var number=obj.length;
        for(var j=obj.length-1;j>=0;j--){
                obj.removeChild(obj.childNodes.item(j));
            }
            for(var i=0;i<a.length;i++){
              var opt=document.createElement("OPTION");
               opt.text=a[i];
               obj.add(opt);
             }
    }
```

在 callback()函数中，调用 show()函数处理服务器端响应信息。在 show()函数中使用 xmlDoc 获取服务器端返回的 XML 文件对象，并使用 xmlDoc 对象的 getElementsByTagName("student")方法获取节点名称为 student 的节点集合对象。在 for 循环中以节点集合的长度为条件，获取节点集对象 content 中每个<student>节点的文本节点，其代码为“y.childNodes[0].data”，并将获取的数据赋值给数组 a[]。最后调用 show1()函数。

在 show1()函数中，使用 document.getElementById("dataMessage")方法获取第二个下拉列表对象 obj，即显示返回数据的对象。创建变量 number 获取第二个下拉列表 obj 的长度。在 for 循环中，使用 removeChild()方法清除下拉列表中已有的选项。在第二个 for 循环中，创建 HTML 节点<opt>，其名称为 OPTION，将数组 a[]中的数据分别赋值给选项文本，即 opt.text。最后将 HTML 节点<opt>追加到 obj 对象中。

（5）实现服务器端程序。服务器端需要根据客户端发送的选项信息返回不同的响应数据。服务器端 PHP 程序如代码 17.29 所示。

代码 17.29　服务器端代码

```
<?php
header('Content-type: text/xml');
$xml="<?xml version='1.0' encoding='GB2312'?>";
$str="5";
$str=$_GET['birthDate'];
$content=$xml."<contents>";
if($str=="5"){
   $content=$content."<student>唐晓阳</student><student>裴亚敏
   </student><student>李现蕾</student><student>任巍</student></contents>";
}else{
   $content=$content."<student>常高洁</student><student>郑海洋</student>
   <student>吕天昊</student><student>李铭</student></contents>";
}
echo $content;
?>
```

上述代码使用$_GET 获取客户端请求信息，并请求信息。输出包含不同数据的 XML 格式文件。

（6）首先保存上述两个文件，然后打开 IE 浏览器，在地址栏中输入“http://localhost/PHP/17.6/DemoSelect.php”，最后单击【转到】按钮，会显示相应页面。在第一个下拉列表中选择不同的选项，会在第二个下拉列表中显示不同的内容，如图 17-12 和图 17-13 所示。

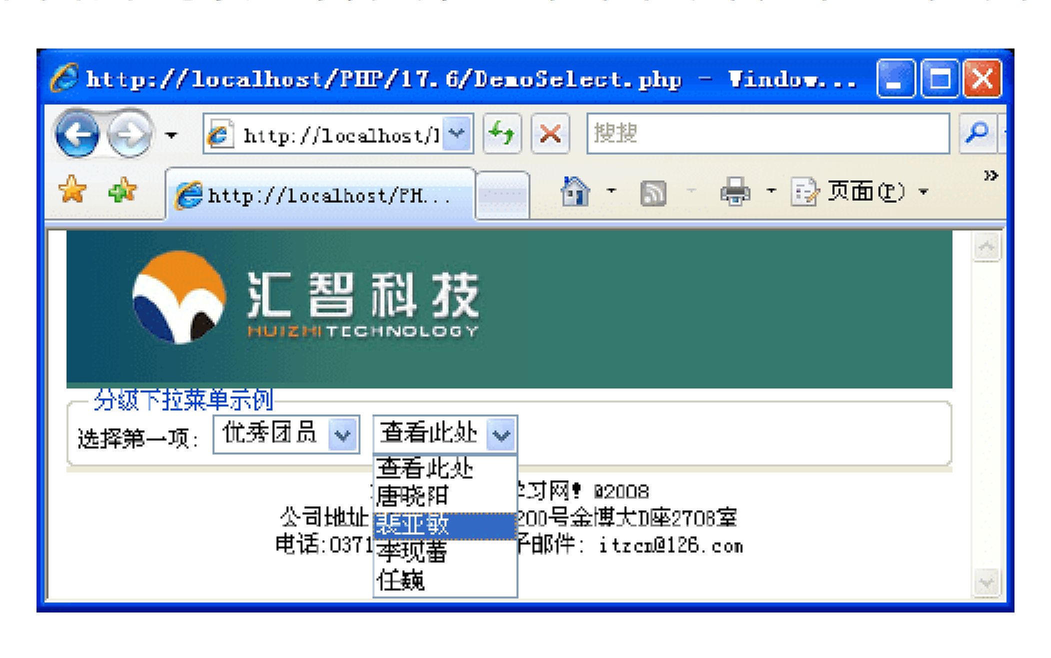


图 17-12　选项一显示

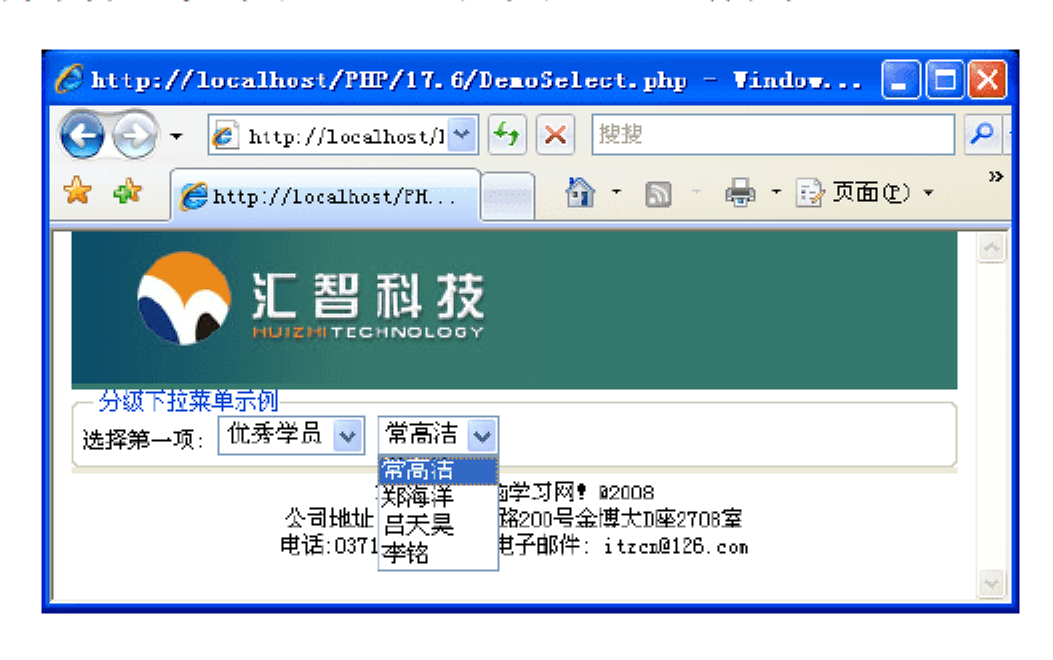


图 17-13　选项二显示

17.7　Ajax+PHP 数据分页显示

在传统的 Web 技术中，分页显示的相关操作都是在服务器端进行。服务器端首先获取客户端的请求页数，然后根据请求页数获取指定的结果集，最后将结果集中的数据返回到客户端。这时返回结果中不但包含了数据，还可能包含其他的数据，如导航栏等。客户端每一次数据更新都会重新打开一个网页，如果网页中包含了很多的 HTML 元素，就会造成网页打开速度较慢等情况。

为了显示部分数据而加载整个页面的数据，显得得不偿失。使用 Ajax 技术可以很好地弥补这些问题，服务器端只传送数据库表中的数据，客户端获取这些数据只更新局部内容，与数据无关的其他元素保持不变。本节将创建一个示例，具体讲解在 PHP 中使用 Ajax 实现数据分页显示。该示例具体开发步骤如下所示。

（1）实现数据库和数据表。为了操作方便和利于读者把握问题实质，本示例采用的数据库和数据表为 17.5 节创建的数据库 books 和表 mybook。

（2）打开记事本，创建 AjaxDb.php 和 AjaxDb1.php 文件，用来实现分页操作的客户端文件和服务器端文件。

（3）打开 AjaxDb.php 文件，首先简单布局该页面，然后实现页面内容显示，其代码如下所示：

```
<body onload="viewpage(1)" style="margin-top:0; font-size:12px">
<table width="100%" cellpadding="0" cellspacing="0">
  <tr> <td background="../image/headerbg.jpg">
   <img src="../image/Header.jpg"></td> </tr>
</table>
<fieldset><legend>数据分页显示</legend>
```

```
<div id="content2"></div>
</fieldset>
</body>
```

在上述代码中，在 body 标签中设置了 onload 属性的值 viewpage(1)，即在页面加载时会调用函数 viewpage()，并传递参数 1。接下来创建了一个名称为 content2 的层。

（4）实现数据请求的发送和响应处理。在 AjaxDb.php 文件中，客户端代码需要向服务器端发送请求的页数，并显示服务器端响应信息。该步骤脚本程序如代码 17.30 所示。

代码 17.30　发送请求页数

```
<script>
function viewpage(p){
    if(window.XMLHttpRequest){
        var xmlReq = new XMLHttpRequest();
    } else if(window.ActiveXObject) {
        var xmlReq = new ActiveXObject('Microsoft.XMLHTTP');
    }
var formData = "page="+p;
xmlReq.onreadystatechange = function(){
        if(xmlReq.readyState == 4){
        document.getElementById('content2').innerHTML=xmlReq.responseText;
    }
}
xmlReq.open("post", "AjaxDb, true);
xmlReq.setRequestHeader("Content-Type", "application/
x-www-form-urlencoded");
xmlReq.send(formData);
return false;
}
</script>
```

在上述代码中，只创建了一个函数 viewpage()，其参数为要求显示的页码。在 viewpage()函数中，首先创建异步传输对象 xmlReq，并创建变量 formData 作为请求信息。接下来设定匿名函数为处理服务器端响应的函数，在匿名函数中将服务器端响应信息在层 content2 中进行显示，如黑体代码。然后使用 open()函数建立一个请求，使用 send()函数把 formData 发送给服务器端。

（5）实现数据分页的服务器端代码。服务器端在获取了客户端传递参数后，根据要求的显示页数，会将相关的数据库记录返回到客户端。该步骤程序如代码 17.31 所示。

代码 17.31　返回数据库数据 PHP 脚本代码

```
<?php
header("Content-Type:text/html;charset=GB2312");
$pagesize=3;                                    //设定 Web 显示的记录数
$db=mysql_connect("localhost","root","0");      //创建数据库连接
```

```
mysql_select_db("books");                           //选择数据库
$result = mysql_query("Select count(DISTINCT id) FROM ".mybook)
                                                    //获取数据库总的记录数
$myrow = mysql_fetch_array($result);
$numrows=$myrow[0];                                 //总的数据记录数
$pages=intval($numrows/$pagesize);                  //求取分页数目
if ($numrows%$pagesize)
$pages++;
if (isset($_POST['page'])){
$page=intval($_POST['page']);                       //获取客户端传递页码
}
else{
$page=1;                                            //设置为第一页
}
$first=1;
$prev=$page-1;
$next=$page+1;
$last=$pages;
$offset=$pagesize*($page - 1);                      //计算记录偏移量
//读取指定记录数
$result=mysql_query("select 'id' , count( * ) from ".mybook." GROUP BY 'id'
order by id desc limit $offset,$pagesize");
$result=mysql_query("select * from ".mybook." GROUP BY `id` order by id desc
limit $offset,$pagesize");
$num = mysql_num_rows($result);
while ($row = mysql_fetch_array($result,MYSQL_NUM)) {
$hotelname[] = $row[0];
$name[]=$row[1];
$author[]=$row[2];
$publisher[]=$row[3];
$isbn[]=$row[4];
$type[]=$row[5];
$price[]=$row[6];
}
//以表格形式输出分页数据
echo "<TABLE style=\"MARGIN-BOTTOM: 20px\" cellSpacing=0 cellPadding=0
width=100% border=1 align=center>\n";
echo "<TBODY>\n";
echo "<th>ID 编号</th><th>名称</th><th>作者</th><th>出版社</th><th>ISBN 号
</th><th>类型</th><th>价格</th>";
for($a=0;$a<$num;$a++)
{
echo "<TR>\n";
echo "<TD style=\"PADDING-TOP: 5px\" align=left>".$hotelname[$a]."</TD>";
echo "<TD style=\"PADDING-TOP: 5px\" align=left>".$name[$a]."</TD>";
echo "<TD style=\"PADDING-TOP: 5px\" align=left>".$author[$a]."</TD>";
echo "<TD style=\"PADDING-TOP: 5px\" align=left>".$publisher[$a]."</TD>";
echo "<TD style=\"PADDING-TOP: 5px\" align=left>".$isbn[$a]."</TD>";
echo "<TD style=\"PADDING-TOP: 5px\" align=left>".$type[$a]."</TD>";
```

```
echo "<TD style=\"PADDING-TOP: 5px\" align=left>".$price[$a]."</TD>";
echo "</TR>";
}
echo "</TR></TBODY></TABLE>\n";
//设定上、下页超级链接
echo "<TABLE style=\"MARGIN-TOP: 30px\" cellSpacing=0 cellPadding=0
width=\"100%\"";
echo "border=0>";
echo "<TBODY><TR><TD colSpan=3 height=20>";
echo "<DIV align=center>";
echo "<P align=left><FONT color=red>第".$page."页/总".$pages."页 |
总".$numrows."条</FONT> | ";
if ($page>1) echo "<a onclick=\"viewpage(".$first.")\" href='#'>首页</a> | ";
if ($page>1) echo "<a onclick=\"viewpage(".$prev.")\" href='#'>上页</a> | ";
if ($page<$pages) echo "<a onclick=\"viewpage(".$next.")\" href='#'>下页</a>
| ";
if ($page<$pages) echo "<a onclick=\"viewpage(".$last.")\" href='#'>尾页</a>";
echo " 转到第 <INPUT maxLength=3 size=3 value=1 name=goto_page> 页 <INPUT
hideFocus onclick=\"viewpage(document.all.goto_page.value)\" type=button
value=Go name=cmd_goto>";
echo "</P></DIV></TD></TR></TBODY></TABLE>";
?>
```

在上述代码中，首先获取当前数据库中总的记录数，并根据总记录数计算出当前可以分多少页。根据从客户端获取的页码计算当前的记录指针，即指针偏移量。创建 SQL 语句，获取指定页码的数据库显示记录，其 SQL 语句为“select 'id' count (*)from ".mybook." GROUP BY `id` order by id desc limit $offset,$pagesize”，此语句表示从指定记录指针处获取指定数目的记录。然后将上述 SQL 语句使用 mysql_query()函数执行，并获取$result 记录集对象；使用 mysql_num_rows()获取显示记录集中的行数。在 for 循环中，将获取的记录集以表格形式输出。

（6）部署和运行。保存这两个文件代码，然后使用浏览器打开 AjaxDb.php，该页面效果如图 17-14 所示。



图 17-14　分页显示数据

17.8　Ajax 自动保存草稿

利用 Ajax 技术可以很好地实现文本编辑时的自动保存，并且还可以实现文本的恢复功能。在 Web 应用程序中，自动保存功能非常有用。例如，浏览器被意外关闭，重新启动后还可以恢复已经编辑一半的文本内容。本节将创建一个 Ajax 自动保存草稿的示例，具体开发步骤如下所示。

（1）打开记事本，创建 DemoSave.php、ajaxrequest.js、autosave.js 和 DemoSave1.php 文件，用来实现客户端程序和服务器端程序。其中 JavaScript 文件 ajaxrequest.js 和 autosave.js 实现异步请求发送和自动保存、保存和恢复功能。

（2）打开 DemoSave.php 文件，首先简单布局一下页面，实现页面内容显示，其代码如下所示：

```
<body style="margin-top:0; font-size:12px">
<table width="100%" cellpadding="0" cellspacing="0">
  <tr> <td background="../image/headerbg.jpg">
   <img src="../image/Header.jpg"></td></tr>
</table>
<fieldset><legend>AJAX 应用之草稿自动保存</legend>
  <table width="100%" class="table4">
    <tr><td align="right">用户名: </td>
      <td><input type="text" name="memName" id="memName" size="20" value=
      "NONAME" disabled="true" />   
      <input onclick="SetAutoSave();" type="checkbox" id="Draft_AutoSave"
      value="1" checked= "true" />是否自动保存</td></tr>
    <tr><td valign="top" align="right">内容: </td>
    <td><textarea cols=40 rows=8 id="message">你编辑的内容将被自动保存,以便恢复
    </textarea></td>
   </tr>
   <tr> <td align="right"><input type="submit" class="Button"  onclick=
   "Save();" value="保存" /></td>
   <td><input type="button" class="Button"  onclick="AutoSaveRestore();"
   value="恢复" /></td></tr>
  </table><div id="AutoSaveMsg"></div></fieldset></body>
```

在上述代码中创建了一个名称为 memName 的文本域，其值为 NONAME，是用户默认名称。接下来创建一个复选框作为文本是否保存的选择，并为该复选框注册了一个 onclick 事件，即当单击此复选框时就会触发 SetAutoSave()函数。然后创建了一个名称为 message 的文本框用来作为文本编辑的组件，还创建了一个名为 AutoSaveMsg 的层作为显示响应信息。最后创建了一个【保存】按钮，并注册了 onclick 事件，即单击此按钮会调用 Save()函数进行保存；创建一个【恢复】按钮，并为该按钮注册了 onclick 事件，即当单击此按钮时就会调用 AutoSaveRestore()函数执行自动保存操作。

（3）引入 js 文件。在 DemoSave.php 文件中并没有包含 js 代码，而是将这些 js 代码封装到不同的 js 文件中，所以在使用这些 js 文件时需要引入到当前文件中，其代码如下所示：

```
<!-- AJAX 类 -->
<script type="text/javascript" src="ajaxrequest.js"></script>
<!-- 自动保存代码 -->
<script type="text/javascript" src="autosave.js"></script>
```

（4）实现 ajaxrequest.js 文件。在 ajaxrequest.js 文件中需要实现异步对象的创建和请求发送。在实现过程中没有采用前面示例实现的过程，而是将这些操作代码封装到一个 JavaScript 类中使用。该步骤程序如代码 17.32 所示。

代码 17.32　脚本程序代码

```
// 创建 AJAX 类 AJAXRequest()
function AJAXRequest() {
    var xmlObj = false;     //创建类成员变量
    var CBfunc,ObjSelf;     //创建类成员变量
    ObjSelf=this;
    try { xmlObj=new XMLHttpRequest; }
    catch(e) {
        try { xmlObj=new ActiveXObject("MSXML2.XMLHTTP"); }
        catch(e2) {
            try { xmlObj=new ActiveXObject("Microsoft.XMLHTTP"); }
            catch(e3) { xmlObj=false; }
        }
    }
    if (!xmlObj) return false;
    this.method="POST";
    this.url;               //创建类成员变量 url
    this.async=true;        //创建类成员变量 async
    this.content="";
    this.callback=function(cbobj) {return;}
    this.send=function() {
        if(!this.method||!this.url||!this.async) return false;
        xmlObj.open (this.method, this.url, this.async);
        if(this.method=="POST") xmlObj.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
        xmlObj.onreadystatechange=function() {
            if(xmlObj.readyState==4) {
                if(xmlObj.status==200) {
                    ObjSelf.callback(xmlObj);
                }
            }
        }
        if(this.method=="POST") xmlObj.send(this.content);
```

```
            else xmlObj.send(null);
        }
    }
```


在类 AJAXRequest()中，首先声明成员变量 xmlObj 和 CBfunc 等。接下来实现异步传输对象的 xmlObj，其实现方式同样为跨浏览器的实现方式。然后声明了几个必需的成员变量，如 url 路径、async 异步传输判断和 content 传输内容。在匿名函数中，将上面声明的变量使用到 xmlObj 对象的 open()调用中。

（5）实现 autosave.js 文件。类 AJAXRequest()创建完成之后就具备了执行 Ajax 程序的基础条件，即实现异步传输对象和发送指定请求。还需要创建 autosave.js 文件代码，用来实现输入内容的获取、不同请求的发送和响应信息处理。该步骤程序如代码 17.33 所示。

代码 17.33　实现异步传输对象和发送指定请求代码

```
// 首先设置全局变量
// 要保存的内容对象 FormContent
var FormContent;
// 显示返回信息的对象
var AutoSaveMsg=document.getElementById("AutoSaveMsg");
// 用户名
var memName=document.getElementById("memName").value;
// 自动保存时间间隔
var AutoSaveTime=10000;
// 计时器对象
var AutoSaveTimer;
// 首先设置一次自动保存状态
SetAutoSave();
// 自动保存函数
function AutoSave() {
     FormContent=document.getElementById("message");
    // 如果内容或用户名为空，则不进行处理，直接返回
    if(!FormContent.value||!memName) return;
    // 创建 AJAXRequest 对象，详细使用见文件开始的链接
    var ajaxobj=new AJAXRequest;
    ajaxobj.url="DemoSave1.php";
    ajaxobj.content="action=AutoSave&memname="+memName+"&postcontent="+
    FormContent.value;
    ajaxobj.callback=function(xmlObj) {
        // 显示反馈信息
        AutoSaveMsg.innerHTML=xmlObj.responseText;
    }
    ajaxobj.send();
}
// 设置自动保存状态函数
function SetAutoSave() {
```

```
    // 是否自动保存?
    if(document.getElementById("Draft_AutoSave").checked==true)
        // 是，设置计时器
        AutoSaveTimer=setInterval("AutoSave()",AutoSaveTime);
    else
        // 否，清除计时器
        clearInterval(AutoSaveTimer);
}
function AutoSaveRestore() {// 恢复最后保存的草稿
     AutoSaveMsg.innerHTML="正在恢复，请稍候……"
     FormContent=document.getElementById("message");
    // 如果用户名为空，则不进行处理，直接返回
    if(!memName) return;
    // 创建 AJAXRequest 对象
    var ajaxobj=new AJAXRequest;
    ajaxobj.url=" DemoSave1.php";
    ajaxobj.content="action=Restore&memname="+memName;
    ajaxobj.callback=function(xmlObj) {
        // 显示反馈信息
     if(xmlObj.responseText!="") {
         // 恢复草稿
         var s=xmlObj.responseText.replace(/^[\n|\r\n]*|[\n|\r\n]*$/g,'');
         //去掉首尾空行
         FormContent.innerText=s;
          // 提示用户恢复成功
        AutoSaveMsg.innerHTML="恢复成功";
        }
    }
    ajaxobj.send();
}
function Save() {
 FormContent=document.getElementById("message");
    // 如果内容或用户名为空，则不进行处理，直接返回
    if(!FormContent.value||!memName) return;
    // 创建 AJAXRequest 对象
    var ajaxobj=new AJAXRequest;
    ajaxobj.url=" DemoSave1.php";
    ajaxobj.content="action=Save&memname="+memName+"&postcontent="+
vFormContent.value;
    ajaxobj.callback=function(xmlObj) {
        // 显示反馈信息
        AutoSaveMsg.innerHTML=xmlObj.responseText;
    }
    ajaxobj.send();
}
```

为了方便读者掌握，代码 17.33 中加入了大量的注释。读者可以参考上面的注释，理解这段 JavaScript 代码。

（6）实现服务器端文件。服务器端文件主要根据客户端发送请求，执行不同的操作，如文件读操作和文件写操作等。该步骤主要程序如代码 17.34 所示。

代码 17.34　服务器端代码

```
<?php
header ("Cache-Control: no-cache, must-revalidate");
   $action=$ POST['action'];
   $memName=$ POST['memname'];
   $PostContent=$_POST['postcontent'];
   echo "$PostContent";
   $filename="c:\aa.txt";
   if($action=="Save"){
     file_put_contents($filename,$PostContent);
   }
   if($action=="Restore"){
       echo  file_get_contents($filename);
   }
   if($action=="AutoSave"){
        file_put_contents($filename,$PostContent);
    }
?>
```

在该文件中，首先使用 header()函数设置页面不允许缓存，并使用$_POST[]获取参数 action、memname 和 PostContent 的参数值。接下来创建目标文件 c:\aa.txt，用来作为读取和写入的文件。如果$action 变量的值为 Save，则将获取的信息写入到 aa.txt 文件中；如果$action 变量的值为 Restore，则读取 aa.txt 中的文本内容；如果$action 变量的值为 AutoSave，则将文件写入到 aa.txt 中。

（7）部署和运行。保存这两个文件代码，然后使用浏览器打开 AjaxDb.php 文件，该页面效果如图 17-15 所示。在【内容】文本框中输入信息后，单击【保存】按钮或者等待一段时间，草稿将会进行保存，并且保存信息会在文本框下面显示，如图 17-16 所示。



图 17-15　草稿自动保存



图 17-16　文本信息保存

（8）单击图 17-15 中的【恢复】按钮可以将 aa.txt 文本文件的信息读取出来，并显示在文本框中。

17.9 信息排序

数据库的数据在显示时，通常都是依据数据输入的先后顺序。但有时在数据显示时，需要数据以另外一种顺序显示，即按照特定字段显示，例如根据图书价格显示图书、根据图书编号显示图书和根据图书分类显示图书等，这样做方便用户对数据进行归类和查询。使用 Ajax 技术可以在同一个页面无刷新的状态下显示不同排序的数据。本节将创建一个示例，该示例具体讲解如何使用 Ajax 实现无刷新信息排序。该示例具体开发步骤如下所示。

（1）实现数据库和数据表。为了操作方便，本示例采用的数据库、数据表还是 17.5 节所使用的 books 数据库和 mybook 数据表。

（2）打开记事本，创建 AjaxData.php 和 AjaxData1.php 文件，用来实现数据排序的客户端程序和服务器端程序。

（3）打开 AjaxData.php 文件，首先为该页面简单布局，然后实现页面显示内容。该步骤页面程序如代码 17.35 所示。

代码 17.35　页面程序代码

```
<body style="margin-top:0; font-size:12px">
<table width="100%" cellpadding="0" cellspacing="0">
  <tr>
   <td background="../image/headerbg.jpg">
   <img src="../image/Header.jpg">
   </td>
  </tr>
</table>
<fieldset><legend>数据表排序</legend>
<table class="table3" width="100%">
<tr><td width="15%">
<select id="xu">
<option value=0>ID 号</option>
<option value=1>图书名称</option>
<option value=2>图书价格</option>
</select>
</td>
  <td><input type=button class="Button" onclick="send()" value="排序"></td>
</tr>
<tr><td colspan="2"><div id="res"></div></td></tr>
</table>
</fieldset>
</body>
```

在代码 17.35 中，创建了一个下拉列表，用来指定数据排序所依据的字段名称。接下来创建了一个按钮，并注册了按钮单击事件，即当按钮被单击时会调用 send()函数。最后创建一个 ID 名称为 res 的层。

（4）实现请求的发送和响应信息的处理。当在下拉列表中选择好数据排序的字段后，单击【排序】按钮就会调用 send()函数发送请求，然后调用相应的函数处理服务器端响应。该步骤主要程序如代码 17.36 所示。

代码 17.36　页面 JavaScript 脚本程序

```
<script type="text/javascript">
var xmlHttp;
function createXMLHttpRequest(){
   if(window.ActiveXObject){
     xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
   else if(window.XMLHttpRequest){
      xmlHttp=new XMLHttpRequest();
   }
}
function send(str){
   createXMLHttpRequest();
   var px=document.getElementById("xu").value;
   var url="AjaxData1.php?px="+px;
   xmlHttp.open("GET",url,true);
   xmlHttp.onreadystatechange=callback;
   xmlHttp.send(null);
}
function callback(){
   if(xmlHttp.readyState==4){
      if(xmlHttp.status==200){
       show();
      }
   }
}
function show(){
   var xmlDoc=xmlHttp.responseXML;
   var yan=xmlDoc.getElementsByTagName("content");
   if(yan.length==0){
         document.getElementById("res").innerHTML="<span style='color:red;
         font-size:15'>要查询的信息不存在</span>";
      }
    else{
   var ta="<table class=\"table4\" width=100% align=center id=ta><th>编号
   </th><th>名称</th><th>作者</th><th>出版社</th><th>ISBN 号</th><th>图书类型
   </th><th>价格</th>";
```

```
        for(var i=0;i<yan.length;i++){
           var y=yan[i];
           ta+="<tr><td>"+y.childNodes[0].firstChild.data+"</td>";
           ta+="<td>"+y.childNodes[1].firstChild.data+"</td>";
           ta+="<td>"+y.childNodes[2].firstChild.data+"</td>";
           ta+="<td>"+y.childNodes[3].firstChild.data+"</td>";
           ta+="<td>"+y.childNodes[4].firstChild.data+"</td>";
           ta+="<td>"+y.childNodes[5].firstChild.data+"</td>";
           ta+="<td>"+y.childNodes[6].firstChild.data+"</td></tr>";
        }
        ta+="</table>";
        document.getElementById("res").innerHTML=ta;
       }
    }
    </script>
```

（5）实现服务器端程序。在 AjaxData.php 文件中，服务器端依据客户端发送的信息创建不同的 SQL 查询语句，并执行这些 SQL 语句，从而获取不同排序的记录集对象。服务器端应用程序如代码 17.37 所示。

代码 17.37　服务器端代码

```
<?php
  header('Content-Type: text/xml');
  $xml="<?xml version='1.0' encoding='GB2312'?>";
  $px=$_GET['px'];
  $px=iconv( 'UTF-8', 'GB2312//IGNORE' , $px);
  $exec="";
  if($px=="0")
     $exec="select * from mybook order by id";
  if($px=="1")
     $exec="select * from mybook order by name";
  if($px=="2")
     $exec="select * from mybook order by price";
  $xml=$xml."<contents>";
  $link=mysql_connect("localhost","root","0");
  mysql_select_db("books");
   $result=mysql_query($exec);
  while($rs=mysql_fetch_object($result))
     {
       $id=$rs->id;
       $name=$rs->name;
       $author=$rs->author;
       $pub=$rs->publisher;
       $isbn=$rs->isbn;
       $type=$rs->type;
```

```
        $price=$rs->price;
      $content=$content."<content><id>".$id."</id><name>".$name.
      "</name><author>".$author."</author><publisher>".$pub."
      </publisher><isbn>".$isbn."</isbn><type>".$type."</type><price>".
      $price."</price></content>";
    }
  $xml=$xml.$content."</contents>";
  echo $xml;
  mysql_close();
  ?>
```

在代码 17.37 中，使用$_GET 获取客户端传递的参数，并将其赋值给变量$px。如果$px 的值为 0，则需要创建依据图书 ID 进行排序的 SQL 语句；如果$px 的值为 1，则需要创建依据图书名称进行排序的 SQL 语句；如果$px 的值为 2，则需要创建依据图书价格进行排序的 SQL 语句。接下来使用创建好的 SQL 语句获取不同排序的记录集对象，并以 XML 文件格式返回。

（6）部署和运行。保存这两个文件代码，然后使用浏览器打开 AjaxData.php 文件。在选择按照 ID 号排序后，单击【排序】按钮页面效果如图 17-17 所示；选择按照图书名称排序，单击【排序】按钮，页面效果如图 17-18 所示；选择按照图书价格排序，单击【排序】按钮，页面效果如图 17-19 所示。



图 17-17　ID 号排序显示



图 17-18　图书名称排序显示



图 17-19　图书价格排序显示

第 5 篇　综合实例篇

第18章 教学视频网站后台管理系统



内容摘要 Abstract

本章将结合实际应用开发一个适合于管理的教学视频网站后台管理系统。该系统主要是对视频、视频分类、视频标签、上传视频、评论管理，以及视频播放列表管理等管理，只有在管理好这些模块的情况下才能很好地管理视频网站。教学视频网站后台管理系统由 PHP+MySQL 完成，并且系统个别功能使用 Ajax 技术处理。本章将详细介绍如何开发视频网站后台管理系统。

学习目标 Objective

- ➢ 整体掌握视频网站后台管理系统
- ➢ 掌握班级管理模块相关功能
- ➢ 掌握如何创建班级
- ➢ 掌握如何实现上传视频功能
- ➢ 掌握如何向班级视频列表中添加视频
- ➢ 掌握使用 Ajax 技术修改视频名称
- ➢ 理解如何实现班级视频列表管理
- ➢ 熟练掌握实现删除视频操作的方法

18.1 系统概述

本章主要讲解如何开发 ITZCN 网络公司旗下的视频教学网站后台管理系统，可以通过访问 http://www.itzcn.com 查看该视频教学网站的前台。视频教学网站后台管理系统主要对视频、视频分类、上传视频，以及班级课程表的管理。该后台管理系统是一个比较大的管理后台，本章主要通过几个模块以及系统整体设计讲解如何实现视频教学网站后台管理系统。

教学视频网站后台管理系统设计的总体目标是实现视频上传、视频删除、评论、班级的管理、准确了解视频播放次数、会员加入班级的人数，以及新建班级课程表和上传新的视频。通过新建班级可以在后台创建一个新的班级课程列表，会员可以通过班级课程列表观看班级课程列表中的视频，以及设置观看班级视频所需要的积分。

视频教学网站后台管理系统的开发比较简单，安全性高、功能全面，并且界面友好。用

户进入系统之后可以很方便地管理网站相关内容。用户登录系统后，首先进入公告管理页面，在该页面上用户可以添加、删除和修改公告相关信息。然后用户可以根据菜单上的导航操作视频教学网站后台管理系统的各个功能。在班级管理模块中，用户可以新建班级、查看和管理班级列表、查看和管理班级课程列表和分享班级等。用户还可以在菜单中单击【上传视频】超级链接为系统添加新的视频，并对这些视频进行管理，例如删除和修改视频名称等。

用户登录视频教学网站后台管理系统后，根据超级管理员分配的权限管理视频教学网站后台管理系统。如果用户登录失败则系统会显示用户登录失败的原因，例如密码错误、无用户名等。上传视频模块，用户可以上传要求上传的视频，如果上传了不符合格式的视频，会提示用户该视频不能上传，在该模块中用户还可以上传视频的缩略图；班级管理功能模块，用户可以根据权限管理该模块，在删除班级时会提示用户是否真的删除该班级，还可以为班级添加视频。在此只扼要介绍这些模块的功能，其如何实现将在下面的章节中详细介绍。图 18-1 为该视频教学网站后台管理系统的流程图。

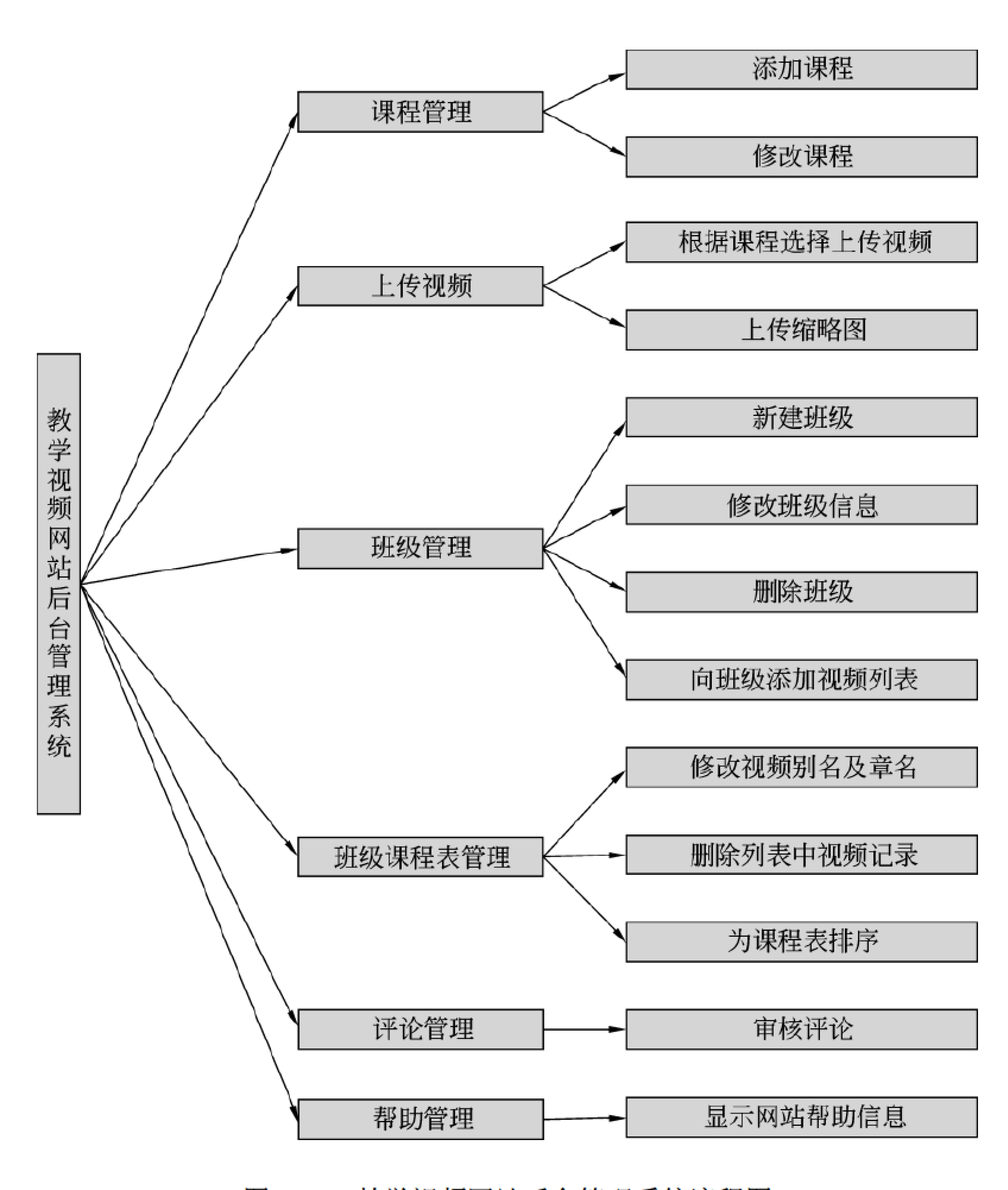


图 18-1　教学视频网站后台管理系统流程图

18.2　数据库设计

在实现教学视频网站后台管理系统之前，首先需要对系统功能进行详细地分析，然后根据上一节作出的系统流程图进行数据库设计。一个好的数据库设计可以加快开发人员的开发速度，并且在开发过程中可以减少开发问题。下面详细介绍本系统涉及到的表。

教学视频网站后台管理系统主要使用到了 7 个表，分别为用户表、用户权限表、课程表、班级表、班级课程列表以及标签表和视频表，其他表会在使用时详细说明其作用。下面详细介绍如何设计这些表。

1. 用户表

用户表（sv_members）主要用来保存用户的相关信息，例如用户名、密码和电子邮件等。该表字段信息如表 18-1 所示。

表 18-1　用户表（sv_members）字段信息

字段	数据类型	允许空	备注
MID	INTEGER	否	主键，用户编号
USERNAME	CHAR(15)	否	用户名
USERPASS	CHAR(32)	否	用户密码
CTIME	INT(10)	否	创建用户时间
EMAIL	CHAR(30)	否	用户密保邮箱
GID	SMALLINT(6)	否	用户权限编号

2. 用户权限表

用户权限表（sv_memgroups）主要是用于存储用户的权限，以及分组。表 18-2 列出了该表字段的详细信息。

表 18-2　用户权限表（sv_memgroups）字段信息

字段	数据类型	允许空	备注
GID	SMALLINT(6)	否	用户权限组编号
SUBJECT	CHAR(30)	否	用户权限组名称
GROUPTYPE	ENUM('system','member','custom')	否	用户权限组类型

3. 课程表

课程表（sv_categories）主要存储上传视频的分类，使用户在组织班级课程表时更容易地从课程中查找到视频文件。表 18-3 列出了课程表字段的详细信息。

表 18-3　课程表（sv_categories）字段信息

字段	数据类型	允许空	备注
CID	INTEGER	否	课程编号
SUBJECT	CHAR(30)	否	课程标题
THUMB	CHAR(128)	否	课程所显示的缩略图

4．班级表

班级表（sv_specials）主要是用于存储班级的相关信息，例如班级名称、班级缩略图和创建者等。表 18-4 列出了该表字段的详细信息。

表 18-4　班级表（sv_specials）字段信息

字段	数据类型	允许空	备注
SPID	INTEGER	否	班级编号
SUBJECT	CHAR(60)	否	班级标题
CID	TINYINT(3)	否	班级分类编号
THUMB	CHAR(16)	是	班级缩略图
AUTHOR	CHAR(16)	否	创建班级者
DESCRIPTION	CHAR(255)	否	班级介绍
TAGS	CHAR(62)	否	班级标签
CTIME	INT(10)	否	班级创建时间
INTEGRAL	INTEGER	是	观看该视频所需积分

5．班级课程列表

班级课程列表（sv_vspecials）主要是用于存储班级的课程列表。班级课程列表按照章节的形式排列，每一个班级有一个课程表，该班级的所有视频会以章节的形式显示到教学视频网站上。表 18-5 列出了该表字段的相关信息。

表 18-5　班级课程列表（sv_vspecials）字段信息

字段	数据类型	允许空	备注
SPID	INTEGER	否	班级编号
VID	INTEGER	否	视频编号
ALIAS	CHAR(60)	否	视频名称别名
ZNAME	VARCHAR(100)	否	视频所在章的章名称

6．标签表

标签表（sv_tags）主要是用于存储班级标签，该表字段信息如表 18-6 所示。

表 18-6　标签表（sv_tags）字段信息

字段	数据类型	允许空	备注
TID	INTEGER	否	标签编号
SUBJECT	CHAR(20)	否	标签标题
CTIME	INT(10)	否	创建时间
SPECIALS	INT(10)	否	班级编号

7．视频表

视频表（sv_videos）主要用于存储视频的相关信息，例如上传者、视频名称、上传时间、

视频地址和缩略图地址等。表 18-7 列出了该表字段的相关信息。

表 18-7　视频表（sv_videos）字段信息

字段	数据类型	允许空	备注
VID	INTEGER	否	视频编号
AUTHOR	CHAR(16)	否	上传者
CID	TINYINT(3)	否	课程编号
SUBJECT	CHAR(60)	否	视频标题
IP	CHAR(15)	否	上传者的 IP
CTIME	INT(10)	否	上传时间
TIMES	SMALLINT(6)	否	视频播放时间
VIDEOSRC	VARCHAR(255)	否	视频地址
PICSRC	VARCHAR(255)	否	缩略图地址

18.3　课程管理

课程管理模块主要是针对视频分类管理。用户按照课程分类向班级中添加视频，这样主要是为了方便用户更好地向班级中查找视频。在课程管理模块中，用户可以添加新的课程、修改课程、删除课程，还可以选择课程的缩略图。本节将主要介绍如何使用 PHP 实现课程管理功能模块。

用户只需单击导航菜单上的【课程设置】超级链接，就可以转到课程设置页面，并且在【课程设置】超级链接上保存了转到页面的相关信息。用户单击【课程设置】超级链接首先转到的页面并不是课程管理页面，而是所有页面共同的转到页面，系统会根据页面传值 script 和 action 来进行判断用户操作的那个页面和做了什么操作。script 值保存了要转到的页面，然后在页面上根据传值 action 判断用户的操作。当用户单击菜单中的超级链接时，首先进入的页面是 admincp.php 页面，用户公共页面判断页面传值，然后再进入用户操作页面。该页面的主要代码如代码 18.1 所示。

代码 18.1　admincp.php 判断传值页面代码

```
<?php
define( "IN_ADMINCP", TRUE );
define( "NOROBOT", TRUE );
include_once( "./inc/inc.common.php" );
include_once( "./admincp/lib/class/class.ui.php" );
require_once( SUPEV_ROOT."./uc_client/client.php" );
$script = empty( $script ) ? "index" : $script;
if ( $script == "logout" )
{
        $sql = "UPDATE {$tablepre}sessions SET admgid=0 WHERE mid='{$member
        ['mid']}'";
        $db->query( $sql );
```

```
        location( "./admincp.php" );
}
if ( $member['gid'] == MEMGRP_GUEST )
{
        show( "请您先到前台进行登录。", "./login.php?action=login&referer=
        ".rawurlencode( "./admincp.php" ) );
}
require_once( LANG_PATH."./admincp/faq_cp.php" );
include_once( LANG_PATH."./admincp.php" );
require_once( SUPEV_ROOT."./admincp/lib/func/func.global.php" );
require_once( SUPEV_ROOT."./admincp/inc/const.php" );
if ( $sess['admgid'] != ADMGID )
{
        include_once( SUPEV_ROOT."./admincp/login.php" );
        exit( );
}
require_once( SUPEV_ROOT."./lib/func/func.settings.php" );
$query = $db->query( "SELECT * FROM {$tablepre}settings" );
while ( $setting = $db->fetch_array( $query ) )
{
        verify_settings( $setting );
        $settings[$setting['var']] = $setting['val'];
}
$page = empty( $page ) ? 1 : max( 1, intval( $page ) );
$scriptarr = array( "index", "header", "menu", "home", "setting", "guest",
"video", "special", "category", "exetension", "cron", "member", "group",
"block", "sysinfo", "reg_insenz", "database", "comment", "announcement",
"union", "infotype", "caches", "words", "ban", "tag", "theme", "log",
"statistical", "vshelp", "copartners", "cop_code", "report", "sitemaps",
"block_infotype", "uc" );
if ( !in_array( $script, $scriptarr ) )
{
        $script = "index";
        $extra = "home";
}
if ( !in_array( $script, array( "index", "header", "menu", "cop_code" ) ) )
{
        cpheader( );
        echo "<table cellspacing=\"0\" cellpadding=\"0\" border=\"0\"
        width=\"100%\">\r\n\t<tr>\r\n\t<td class=\"mainarea\">";
}
include_once( SUPEV_ROOT."./admincp/".$script.".php" );
if ( !in_array( $script, array( "index", "header", "menu", "cop_code" ) ) )
{
        echo "</td></tr></table>";
```

```
        cpfooter( );
}
?>
```

在代码 18.1 中，加粗部分的代码表示该 PHP 文件包含 client.php 文件，client.php 文件为一个类文件，在以后的代码中使用到类文件中的方法时会逐一讲解，在此就不再赘述。

在代码 18.1 中，首先判断了用户是否登录，如果用户没有登录便会返回登录页面。还使用到了 sv_session 表，该表的主要作用是存储用户登录后的用户名信息，当用户再次登录系统时会随时更新其值。然后使用变量$scriptarr 保存该后台系统涉及到的所有页面，接下来判断用户单击的链接中 script 所保存的值是否在该数组中。如果不存在就转到后台管理系统主页，变量 script 值如果为“"index", "header", "menu", "cop_code"”4 个值中的任何一个值就创建 Table 表格。

菜单中的【课程设置】超级链接的代码如下所示：

```
<li><a href="admincp.php?script=category&action=list" target="main">课程设置
</a></li>
```

由上述代码可知，【课程设置】超级链接 script 传值为 category，因此代码 18.1 所转到的页面应该是 category.php 页面。转到 category.php 页面后，程序会根据 action 的值判断页面显示的样式。例如 action 值为 list，category.php 页面则会将课程显示到页面上，如果 action 为其他值该页面会进行相应的操作。当用户单击该超级链接时，程序会进入 action 为 list 的程序中执行，从而显示课程列表。显示课程列表的 PHP 程序如代码 18.2 所示。

代码 18.2　显示课程列表代码

```
if ( $action == "list" )
{
        if ( !submitcheck( "submitbtn" ) )
        {
                $uilist = new ui_list_class( );
                $uilist->name = "课程管理";
                $uilist->header = array(
                        "cid" => model_header( "", "10%", M_CHECKBOX, "KEY" ),
                        "displayorder"=>model_header("排序","10%",M_TEXT),
                        "thumb" => model_header( "课程图片", "20%", M_LABEL ),
                        "subject" => model_header( "课程名称", "20%", M_TEXT ),
                        "enable_nav" => model_header( "是否在导航显示", "20%",
                        M_TEXT ),
                        "insenz_category" => model_header( "Insenz 课程",
                        "15%", M_LABEL ),
                        "oprow" => model_header( "操作", "20%", M_LABEL )
                );
                $sql = "select * from {$tablepre}categories order by
                displayorder ";
                $res = $db->query( $sql );
```

```
            $rowarr = array( );
            while ( $cate = $db->fetch_array( $res ) )
            {
                    $cate['oprow'] = "<a href=\
                    "?script=category&action=edit&cid=".$cate[cid]."\">
                    编辑</a>";
                    $cate['insenz_category'] = $video_categories
                    ['categorys'][$cate['insenz_category']];
                    $cate['thumb'] = "<img src=\"
                    ".CATETHUMBPATH.$cate['thumb']."\" title=
                    \"".$cate['subject']."\" alt=\"".$cate['subject'].
                    "\" onerror=\"this.src='".CATENOTHUMB."';\" />";
                    $rowarr[] = $cate;
            }
            $uilist->data =& $rowarr;
            $uilist->opt = array(
                    "name" => model_optgroup( "operation", "" ),
                    "values" => array(
                            "update" => model_optitem( "更新" ),
                            "del" => model_optitem( "删除", M_CANCELBOX )
                    )
            );
            $uilist->show( );
    }
```

当用户在 category.php 页面上无任何操作时，进入代码 18.2 中的 if 语句。在代码 18.2 中，首先实例化了一个 ui_list_class 类，该类在 admincp 文件夹的子文件夹 class 中的 class.ui.php 文件中。ui_list_class 类的主要作用是根据得到的值输出页面，然后为 ui_list_class 类中的变量 name 和 header 赋值，这些值能直接影响显示列表的布局。

接下来从数据库中获取课程的相关信息，保存到 rowarr[]数组中，然后将该数组赋值给 ui_list_class 类的变量 data，最后调用该类中的 show()方法输出该课程列表。由于该方法比较大，下面分段介绍该方法。首先展示输出 header 部分代码，该段程序如代码 18.3 所示。

代码 18.3　输出课程列表方法

```
function show( )
        {
            $multipage= $GLOBALS['multipage'];
            echo LF."<form action=\"".$this->url."\" method=\"post\"
            onsubmit=\"return submitcheck(this);\">";
            showtype( $this->name, "top", 99, 1, "showlist" );
            echo "<tr class=\"category\">";
            foreach ( $this->header as $key => $head )
            {
                    if ( strtolower( $head['key'] ) == "key" )
```

```
{
        $primarykey = $key;
}
else if ( $head['key'] )
{
        $size = "size=".$head['key'];
}
if ( $head['type'] == M_CHECKBOX &&
strtolower( $head['key'] ) == "key" )
{
        $listarr[$key] = "<td width=".$head
        ['width']."><input class=checkedbox name=
        delitems[] type=checkbox value= \
        $row[".$key."] ></td> ";
}
else if ( $head['type'] == M CHECKBOX )
{
        $listarr[$key] = "<td width=".
        $head['width']."><input class=checkedbox
        name=items[\$row[\$primarykey]][".$key."]
        type=checkbox value=1 \$row
        [".$key."_checked]></td> ";
}
else if ( $head['type'] == M_TEXT )
{
        $listarr[$key] = "<td width=".
        $head['width']."><input
        name=items[\$row[\$primarykey]][".$key."]
        value= \$row[".$key."] ></td>";
}
else if ( $head['type'] == M_TTEXT )
{
        $listarr[$key] = "<td width=
        ".$head['width']."> \$row[".$key."] </td>";
}
else if ( $head['type'] == M_COLORSEL )
{
        $listarr[$key] = "<td width=".$head
        ['width']."><input id=items[\$row[\
        $primarykey]][".$key."] name=items[\$row
        [\$primarykey]][".$key."] ".$size."
        onchange=\\\"./js/admincp/
        getcolor.htm?items[\$row[\$primarykey]]
        [".$key."];showMenu(\\\"items
        [\$row[\$primarykey]][".$key."]\\\");\\\"
```

```
                        value= \$row[".$key."]  ></td>";
                }
                else if ( $head['type'] == M_TCALENDAR )
                {
                        $listarr[$key] = "<td width=".$head
                        ['width']."><input id=items
                        [\$row[\$primarykey]][".$key."]
                        name=items[\$row[\$primarykey]][".$key."]
                        onclick=\\\"showcalendar(event, this)\\\"
                        value=\\\" \$row[".$key."]\\\"  ></td>";
                }
                else
                {
                        $listarr[$key] = "<td
                        width=".$head['width']."> \$row[".$key."]
                        </td>";
                }
                if ( strtolower( $head['key'] ) != "key" )
                {
                        echo "<td width=\"".$head['width']."\
                        ">".$head['tag']."</td>";
                }
                else if ( $head['type'] == M_CHECKBOX )
                {
                        echo "<td width=\"".$head['width']."\">
                        <input name=\"chkall\" class=\"checkedbox\"
                        onclick=\"checkall(this.form, 'delitems')\"
                        type=\"checkbox\">全选</td>";
                }
                else
                {
                        echo "<td width=\"".
                        $head['width']."\">ID</td>";
                }
        }
```

在代码 18.3 中，根据判断数组 type[]所指向的值输出相应的代码。该段代码比较简单，在此就不再做详细解释。下面介绍输出课程列表内容的代码，如代码 18.4 所示。

代码 18.4　输出课程列表内容部分代码

```
$liststr = "";
foreach ( $listarr as $listtemp )
{
        $liststr .= $listtemp;
}
```

```
    echo "</tr>";
    if ( is_array( $this->data ) )
    {
            foreach ( $this->data as $row )
            {
                    eval( "\$liststrnew = \"{$liststr}\";" );
                    echo "<tr>".$liststrnew."</tr>";
            }
    }
```

在代码 18.4 中，将从数据库中得到的课程信息列出到界面上。下面介绍输出该页面上的下拉菜单和【提交】按钮的代码，如代码 18.5 所示。

代码 18.5　输出下拉菜单和【提交】按钮程序

```
    if ( !empty( $this->opt ) )
    {
            $opsidestr_sub = _multi_select( $this->opt );
            echo "<table cellspacing=\"0\" cellpadding=\"0\"
            width=\"99%\"><tr>";
            echo "<td width=\"\" align=\"left\">{$opsidestr_sub}
            </td><td align=\"right\">{$multipage}</td>";
            echo "</tr></table>";
            echo "</div>";
    }
    if ( $this->submitbtn == true )
    {
            echo "<center><input id=\"submitbtn\" class=
            \"button_disabled\" type=\"submit\" name=
            \"submitbtn\" disabled value=\"提交\"></center>";
    }
    echo "</form>";
    if ( $this->submitbtn == true )
    {
            echo LF."<script language=\"javascript\" type=
            \"text/javascript\">";
            echo "swapopside(\"op_1_\", \"\", \"opsidediv_1\",
            this);";
            echo "if(\$('operation').value=='')
            {showsubmit(false);is_confirm = false;}else
            if(\$('operation').value=='del'){is_confirm =
            true;}";
            echo "</script>";
    }
```

在代码 18.5 中，根据页面要求输出了下拉菜单和【提交】按钮，并且在输出页面上添加

了 JavaScript 语言函数。当无选择值时按钮为不可用，如果有选择值则显示【提交】按钮。当对列表进行操作时，程序会进入代码 18.2 中的 else 语句中进行相应的操作，如代码 18.6 所示。

代码 18.6　更新和删除列表中项代码

```
else
{
        $table = "categories";
        if ( $operation == "update" )
        {
                foreach ( $items as $lineid => $line )
                {
                        $sqltemp = $comma = "";
                        foreach ( $line as $keys => $lists )
                        {
                                $sqltemp .= $comma.$keys."=
                                '".$lists."'";
                                $comma = ",";
                        }
            $sql = "update {$tablepre}{$table} set ".$sqltemp." Where
            cid='{$lineid}'";
                        $db->query( $sql );
                }
        }
        else if ( $operation == "del" )
        {
                if ( !$confirmed )
                {
                }
                if ( !empty( $delitems ) )
                {
                        $comma = "";
                        foreach ( $delitems as $delid )
                        {
                                $delids .= $comma.$delid;
                                $comma = ",";
                        }
                $sql = "delete from  {$tablepre}{$table}  where cid in
                ({$delids})";
                        $db->query( $sql );
                }
        }
        updatesyscache( "common" );
        cpmsg( "操作成功。", "?script=category&action=list" );
}
```

此时课程管理页面应用程序就成功地完成了。当链接到该页面时，页面上就可以输出课程的相关信息，并且可以在该页面删除和更新课程信息。在浏览器中打开课程管理页面，页面效果如图 18-2 所示。

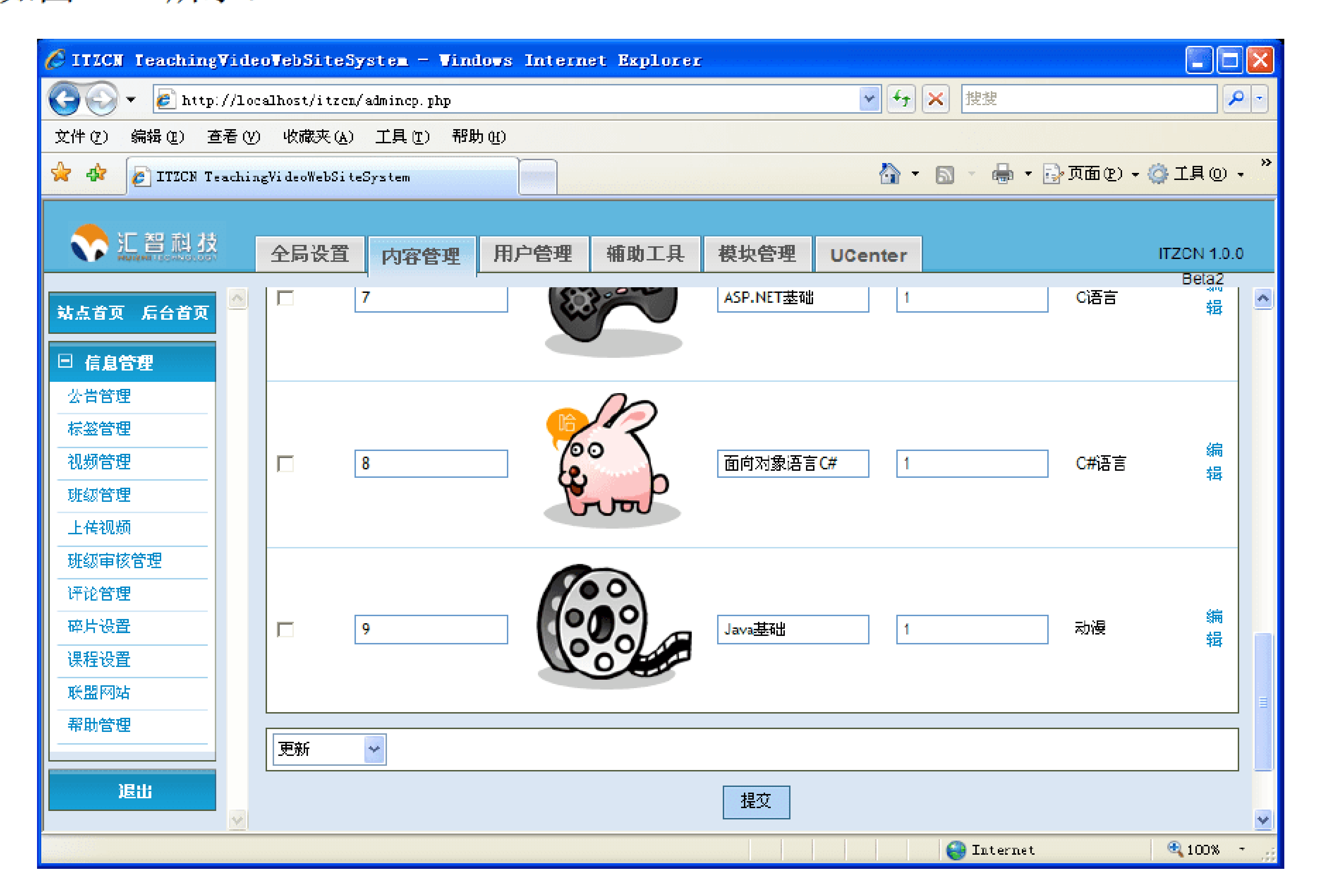


图 18-2　课程设置页面效果图

18.4　上传视频

上传视频是教学视频网站后台管理系统的核心部分。如果不能上传视频，整个视频网站则没有任何意义，这是因为如果没有上传视频功能就不能更新网站中的视频。本节主要介绍如何实现上传视频模块。

上传视频模块页面和课程页面使用不同的方法实现，上传视频页面使用模板实现页面。在页面模板上使用了标题头，由于该页面使用到了 Ajax 技术，因此在该页面的标题头还加载了 Ajax 标题头。代码 18.7 展示了这两个标题头的代码。

代码 18.7　标题头源程序

```
<?php exit();?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=$charset" />
<meta name="keywords" content="$settings[seo keywords]" />
<meta name="description" content="$metadescription
$settings[seo_description]" - SupeV />
```

```
<title>$seo_title</title>
<!--{if isset($invspacecp)}-->
<link href="{THEME_PATH}css/vspacecp.css" rel="stylesheet" type="text/css" />
<!--{else}-->
<link href="{THEME_PATH}css/supev.css" rel="stylesheet" type="text/css" />
<!--{/if}-->
<script>
    $jscode
</script>
<script src="./js/common.js"></script>
<script src="./js/ajax.js"></script>
<!--{if $member['mid'] && !isset($cookie['checkpm'])}-->
<script type="text/javascript" src="./pm.php?checknewpm" defer="true">
</script>
<!--{/if}-->
<!--{if isset($ajaxpage)}-->
    {template ajax_page}
<!--{/if}-->
<!—Ajax 表头-->
{eval ob_end_clean();}
{eval ob_start();}
{eval @header("Expires: -1");}
{eval @header("Cache-Control: no-store, private, post-check=0, pre-check=0,
max-age=0", FALSE);}
{eval @header("Pragma: no-cache");}
{eval @header("Content-type: application/xml; charset=$charset");}
{eval echo '<?xml version="1.0" encoding="'.$charset.'"?>';}
<root><![CDATA[
```

代码 18.7 展示了标题的代码，在页面上直接引用这个文件就可以。上传视频页面引用这两个标题头的代码的方法如下所示：

```
<?php exit();?>
<!--{if isset($inajax)}-->
{template ajax_header}
<!--{else}-->
{template addvideoheader}
<!--{/if}-->
```

以上介绍了如何设计页面标题，接着介绍上传视频页面的设计。页面的设计比较简单，在此只介绍页面上在选择课程时绑定课程列表的方法。上传视频页面上显示课程列表的页面源程序如代码 18.8 所示。

代码 18.8　显示课程列表源程序

```
<ul>
```

```
        <li>
          <select name="cid" style="width:200px" id="category_{$category[cid]}">
            <!--{loop $cache['categories'] $category}-->
            <option value="$category[cid]">$category[subject]</option>
            <!--{/loop}-->
          </select>
        </li>
    </ul>
```

在代码 18.8 中，首先从数组$category[]中读取课程相关信息，$category[]数组是从数据库中读取的数据。代码中加粗的部分表示一个循环，该循环执行方式跟 foreach 循环相似，在此就不再多做介绍。接下来介绍【上传文件】文本框，该文本框旁边有一个按钮，该按钮的 onclick 事件引用脚本程序打开 admin_swfup.php 文件。admin_swfup.php 文件用于视频文件上传，在该文件中限制了上传文件的上传格式，以及上传文件的大小。本节主要讲解的是如何向数据库中保存视频信息，在此就不再赘述如何上传文件，读者可以参考光盘中的源程序。

接下来介绍用户上传视频并添加相关信息后，PHP 如何处理该提交程序。从该上传视频源代码处可以很清楚的看出，该功能的 action 值为 upload，因此在用户提交该页面时 PHP 程序会进入 action 值为 upload 的 if 判断语句中。在该判断语句中，首先判断当前时间是否可以上传视频，然后再判断用户组是否可以上传视频，如果用户所在用户组可以上传视频，接下来判断用户当日上传视频是否超出上传视频上限，这些程序如代码 18.9 所示。

代码 18.9　判断用户是否可以上传视频程序

```
    if ( $action == "upload" )
    {

            if ( intval( $confineslot['postbanperiods'] ) == ON )
            {
                    show( "对不起，当前时间段不允许上传视频。", "./vspacecp.
                    php?script=home" );
            }
            else if ( $grouppurview['allowupload'] == OFF )
            {
                    show( "对不起，您所在的用户组不允许上传视频。", "./vspacecp.php?
                    script=home" );
            }
            else if ( $grouppurview['uploadnumper'] != 0 )
            {
                    $timelimit = smktime( sdate( $timestamp, "Y-m-d" ) );
                    $sql = "SELECT COUNT(*) FROM {$tablepre}videos WHERE
                    authorid='{$member['mid']}' AND ctime>'{$timelimit}'";
                    $todayvideos = $db->sresult( $sql );
                    if ( $grouppurview['uploadnumper'] <= $todayvideos )
                    {
```

```
                show( "您今日上传视频的数量已经达到了上限。", "./
                vspacecp.php?script=addvideo" );
            }
        }
```

经过代码 18.9 对用户是否可以上传视频的判断，如果用户可以上传视频，用户进入上传视频的 PHP 程序中。在此只列出保存视频信息的代码，如代码 18.10 所示。

代码 18.10　保存视频程序代码

```
$paramarr = array(
        "tags" => $tags,
        "ivid" => $ivid,
        "cid" => $cid,
        "user" => array(
                "myid" => $member['mid'],
                "username" => $member['username']
        ),
        "subject" => $subject,
        "times" => $times,
        "isorig" => $isorig,
        "description" => $description,
        "allowgrade" => $allowgrade,
        "allowcomment" => $allowcomment,
        "infotypeid" => 0,
        "isup" => VIDEO_ISUP,
        "autoplay" => VIDEO_AUTOPLAY_TRUE,
        "publish" => $publish_video,
        "videosrc" => $videosrc,
        "picsrc" => $picsrc
);

$mvideo->call( "on_create", $paramarr );
$vid =$GLOBALS['vid'];
if ( $mvideo->errno < 0 )
{
        switch ( $mvideo->errno )
        {
        case ERROR_PERIODS :
                addshow( $mvideo->errmsg, "./vspacecp.php?script=
                video&action=list" );
                break;
        case ERROR_TAG :
        addshow( $mvideo->errmsg, "./vspacecp.php?
        script=video&action=list" );
                break;
```

```
            case ERROR_PUBLISH :
                    addshow( $mvideo->errmsg, "./vspacecp.php?
                    script=video&action=list" );
                break;
        default :
                addshow( $mvideo->errmsg, "./vspacecp.
                php?script=video&action=list" );
        }
    }
```

在代码 18.10 中，首先获取用户输入的视频的相关信息，然后存放到数组中，最后调用 on_create()方法，使用该方法保存视频信息。on_create()方法如代码 18.11 所示。

代码 18.11　on_create()方法

```
    function on_create( $mparam, $mask_publish, $mask, $err )
        {
                $cache=$GLOBALS['cache'];
                $grouppurview =$GLOBALS['grouppurview'];
                $settings =$GLOBALS['settings'];
                $confineslot =$GLOBALS['confineslot'];
                $onlineip =$GLOBALS['onlineip'];
                $timestamp =$GLOBALS['timestamp'];
                $categoryarr =$GLOBALS['categoryarr'];
                if ( empty( $mparam['videosrc'] ) ||empty( $mparam['picsrc'] )  ||
                empty( $mparam['tags'] ) || empty( $mparam['user']['myid'] ) ||
                empty( $mparam['user']['username'] ) || !module_
                cls::validate_uint($mparam['cid'])||empty($mparam['subject'])
                || empty( $mparam['autoplay'] ) )
                {
                        $err['errno'] = ERROR PARAM;
                        $err['errmsg'] = $this->lang[ERROR_PARAM];
                        return ERROR_PARAM;
                }
                $tags = getvalid_tags( $mparam['tags'] );
                if ( empty( $tags[OPTAGS] ) )
                {
                        $err['errno'] = ERROR_TAG;
                        $err['errmsg'] = $this->lang[ERROR_TAG];
                        return ERROR_TAG;
                }
                $arr = array(
                        "vid" => "",
                        "ivid" => $mparam['ivid'],
                        "authorid" => intval( $mparam['user']['myid'] ),
                        "author" => dhtmlspecialchars( $mparam
                        ['user']['username'] ),
                        "cid" => intval( $mparam['cid'] ),
                        "infotypeid" => $mparam['infotypeid'],
                        "tags" => implode( ",", $tags[OPTAGS] ),
```

```
                "subject" => dhtmlspecialchars( $mparam['subject'] ),
                ……
                "ip" => $onlineip,
                "publish" => intval( $mparam['publish'] ),
                "videosrc"=>$mparam['videosrc'],
                "picsrc"=>$mparam['picsrc']
        );
        $this->db->insert( $this->tablepre."videos", $arr );
        $vid = $this->db->insert_id( );
        $arr = array(
                "vid" => $vid,
                "description" => dhtmlspecialchars
                ( $mparam['description'] ),
                "allowcomment" => intval( $mparam['allowcomment'] ),
                "allowgrade" => intval( $mparam['allowgrade'] ),
                "autoplay" => $mparam['autoplay']
        );
        $this->db->insert( $this->tablepre."videofields", $arr );
        @tags addnew( $tags, array(
                "mid" => $mparam['user']['myid'],
                "vid" => $vid,
                "cid" => $mparam['cid']
        ) );
        if ( $publish_video == PUBLISH_AUDITING )
        {
                $err['errno'] = ERROR_PUBLISH;
                $err['errmsg'] = $this->lang[ERROR_PUBLISH];
                return ERROR_PUBLISH;
        }
        @update_member_nums( $mparam['user']['myid'], "videos" );
        @updateviews( "sub member", $mparam['user']['myid'], true, false,
        $vid );
        @updatecredits( $mparam['user']['myid'], $settings
        ['credit_rule']['publishvideo'] );
        $err['errno'] = MSG_SUCCESS;
        $err['errmsg'] = $this->lang[MSG_SUCCESS];
        $GLOBALS['vid']=$vid;
        return MSG_SUCCESS;
    }
```

在代码 18.11 中，$arr[]数组的赋值中有省略的代码（此段代码可以参考光盘中的源程序），然后编写插入数据库的 SQL 语句。on_create()方法会返回执行 SQL 语句的执行结果。如果用户在页面上选择了视频要加入的班级，在代码 18.10 中还需要添加一个判断语句，判断用户是否选择了视频加入的班级。如果没有选择直接结束该段代码，否则执行加入班级的操作。该判断语句如代码 18.12 所示。

代码 18.12　执行插入班级课程列表代码

```
    if ( !empty( $spid ) )
    {
```

```
            $mvideo->call( "on_tospecials", array(
                    "vid" => $vid,
                    "spid" => $spid,
                    "myid" => $member['mid']
            ) );
            if ( $mvideo->errno < 0 )
            {
                    addshow( "增加到指定班级失败。", "./vspacecp.
                    php?script=video&action=list" );
            }
    }
    $mmember = new module_cls( "member" );

    $mmember->call( "on_updatelast", array(
            "myid" => $member['mid'],
            "limit" => 3
    ) );

    if ( $publish video == PUBLISH ACCESS )
    {
            addshow( "视频上传成功。", "./vspacecp.php?script=
            addvideo&action=list" );
    }
    else if ( $publish_video == PUBLISH_AUDITING )
    {
            addshow( "视频上传成功，该视频需要审核。", "./vspacecp.php?
            script=addvideo&action=list" );
    }
```

在代码 18.12 中，运用到了 on_tospecials()方法。由于该方法比较简单，在此就不再多作讲解。运行程序，该页面效果如图 18-3 所示。



图 18-3　上传视频页面效果图

18.5 班级管理模块

在班级管理模块中，用户可以创建班级、删除班级、修改班级信息、组织课程表，以及管理班级课程表等。该模块是教学视频网站后台管理系统的另外一个核心。该模块的重要作用就是为网站客户布置课程，让网站客户更轻松、更方便地在教学视频网站上找到自己要学习的课程内容。本节主要介绍如何实现班级管理模块。

18.5.1 创建班级

创建班级主要是为了组织视频，以及方便网站客户加入班级学习视频。创建班级只需要填写一些相关信息和加入该班级所需要的积分。创建班级模块的实现也是通过模板实现界面，然后在后台通过判断 action 值，进行处理用户操作。

创建班级模块的实现首先要创建一个模板，然后在该模板页面 form 中绑定数据库中的课程信息数组，最后在该页面中添加一些输入控件，让用户输入班级相关信息，例如班级标题、积分和详细介绍等。由于该页面布局比较简单，在此就不再赘述。下面介绍当用户填写完整信息时，单击【确定提交】按钮所触发的 JavaScript 脚本验证程序，如代码 18.13 所示。

代码 18.13 JavaScript 脚本验证程序

```
function createspecial(obj) {
    if(trim(obj.subject.value) == '') {
        obj.subject.focus();
        alertmenu('班级标题不能为空。', 'subject');
        return false;
    }
    if(trim(obj.integral.value) == '') {
        obj.integral.focus();
        alertmenu('班级积分不能为空。', 'integral');
        return false;
    }
    if(trim(obj.tags.value) == '') {
        obj.tags.focus();
        alertmenu('班级标签不能为空。', 'tags');
        return false;
    }
    if(trim(obj.description.value) == '') {
        obj.description.focus();
        alertmenu('班级介绍不能为空。', 'description');
        return false;
    }
    obj.submit();
}
```

在代码 18.13 中，在用户提交信息到服务器之前，首先验证了用户输入的信息是否合法。如果不合法就返回；如果用户按要求填写相关信息，JavaScript 脚本程序会将这些信息提交到

服务器处理。在创建班级页面的源代码中很容易找到该功能的 action 值为 add，当用户单击【确定提交】按钮将信息提交到服务器时，应用程序会转到 action 值为 add 的判断语句中执行程序。在该判断语句中，首先判断该用户是否可以创建班级，此部分代码可以参考代码 18.9。下面列出创建班级的 PHP 程序，具体如代码 18.14 所示。

代码 18.14　创建班级 PHP 程序

```
if ( !submitcheck( "createsubmit" ) )
{
        $cid = isset( $cid ) ? $cid : "";
        $subject = isset( $subject ) ? $subject : "";
        $description = isset( $description ) ? $description : "";
        $tags = isset( $tags ) ? $tags : "";
        $status = "nosubmit";
        if ( $op == "pltosp" && empty( $subject ) )
        {
                $timestr = sdate( $timestamp, "Y-m-d_G-i-s" );
                $subject = "点播单".$timestr;
        }
}
else
{
        $mparam = array(
                "subject" => $subject,
                "cid" => $cid,
                "tags" => $tags,
                "op" => $op,
                "myid" => $member['mid'],
                "admgid" => $member['admgid'],
                "username" => daddslashes( $member['username'], true ),
                "description" => $description,
                "integral"=>$integral
        );
        $mspecial->call( "on_create", $mparam );
        $spid = $db->sresult( "SELECT spid FROM {$tablepre}specials WHERE
        subject='".$subject."'\r\n\t\t\t\t     AND authorid = '".
        $member['mid']."' LIMIT 1" );
        if ( $isfeed )
        {
                        $param_arr = array(
                        "spid" => $spid,
                        "myid" => $member['mid'],
                        "username" => $member['username'],
                        "subject" => $subject,
                        "description" => $description
```

```
                );

                $mfeed = new module_cls( "feed" );
                $mfeed->call( "push_special", $param_arr );
            }
            show( $mspecial->errmsg, "./vspacecp.php?
            script=specials&action=list" );
        }
```

在代码 18.14 中，首先获取了界面用户输入的值，然后通过对象$mspexial 执行 call()方法，最后再通过 call()方法执行创建班级的 on_create()方法，此时就成功地创建了一个新班级。运行程序，在创建班级页面创建一个班级，页面效果如图 18-4 所示。



图 18-4　页面效果图

18.5.2　管理班级

管理班级模块展示的是班级列表，所有的班级都能在该页面上显示。管理班级页面还可以删除班级、分享班级、编辑班级、创建新班级和添加视频等。本小节讲解如何实现班级管理模块，主要讲解的是删除班级和分享班级的实现，其他功能的实现将在下一小节中讲解。

在显示班级列表页面中，首先获取从数据库中读取的班级信息，然后在页面上通过使用循环读取班级相关信息，并让班级以列表的形式显示到页面上。班级列表显示页面的 action 值为 list。运行教学视频后台管理系统，当用户单击【班级管理】超级链接时，系统会自动转到 action 为 list 的程序语句中执行查询班级信息的任务。实现该功能的程序语句如代码 18.15 所示。

代码 18.15　获取班级列表

```
else if ( $action == "list" )
{
        $orderarr = array( );
        $mspecial->addorder( $orderarr, "utime" );
        $mparam = array(
                "myid" => $member['mid'],
                "start" => $rowmax * ( $page - 1 ),
                "limit" => $rowmax,
                "order" => $orderarr,
                "fmttime" => "Y-m-d H:i:s"
        );
        $mres = $mspecial->call( "qy_user", $mparam, PUBLISH_ALL ^ PUBLISH_
        RECYCLE );
        if ( $mspecial->errno != MSG_SUCCESS )
        {
                show( $mspecial->errno, "./vspacecp.php" );
        }
        $special = $mres['data'];
        $multipage = multi( $mres['count'], $rowmax, $page, "./vspacecp.
        php?script=specials&action=list", 0 );
}
```

在代码 18.15 中，首先获取了班级信息，然后在班级列表界面通过循环显示。运行程序，单击【班级管理】超级链接转到该页面，页面效果如图 18-5 所示。



图 18-5　班级管理页面效果图

在页面显示列表后，接着实现管理班级的相关功能。首先介绍删除班级功能的实现，图18-5中的【新建班级】和【删除班级】按钮在一个单独的form表单中，并且这两个按钮的action值不同。【删除班级】按钮的action值为del，当单击【删除班级】按钮系统会自动提交数据到action值为del的判断语句中。在提交之前JavaScript脚本程序会验证用户是否选择所要删除的班级，如果没有选中则会给用户一个提示。该脚本程序如代码18.16所示。

代码 18.16　JavaScript 脚本验证用户是否选择要删除的班级

```
function muldelspecial(obj, btnid) {
    var checknum = 0;
    for( var i = 0; i < obj.elements.length; i++ ){
        var e = obj.elements[i];
        if(e.type == "checkbox" && e.name == "spid[]" && e.checked ==
        true) {
            checknum ++;
        }
    }
    if(checknum > 0) {
        if(confirm('你确定要删除选中的班级吗？')) {
            obj.submit();
        } else {
            return false;
        }
    } else {
        alertmenu('请先选择要操作的班级。', btnid);
    }
}
```

经过界面脚本程序验证，如果验证通过系统会自动转到PHP处理程序中。在PHP删除程序中，首先判断用户选择的班级是否存在，如果存在则进行删除处理程序，否则返回。删除班级的处理程序如代码18.17所示。

代码 18.17　处理删除班级 PHP 程序

```
else if ( $action == "del" )
{
        if ( empty( $spid ) )
        {
                show( "指定班级不存在或者已经被管理员删除。", "./vspacecp.
                php?script=specials&action=list" );
        }
        $mparam = array(
                "spid" => $spid,
                "mid" => $member['mid']
        );
        $mspecial->call( "on_del", $mparam, PUBLISH_ALL ^ PUBLISH_RECYCLE );
```

```
        show( $mspecial->errmsg, "./vspacecp.php?
        script=specials&action=list" );
    }
```

在代码 18.17 中实现了删除班级的功能，接下来实现分享班级的功能。分享班级实际上是使用 JavaScript 脚本程序实现的，首先使用脚本获取班级的连接，然后令用户复制到剪切板上。该脚本程序如代码 18.18 所示。

代码 18.18　实现分享班级功能代码

```
function aj_form(e, ctrlid, divclass, lefttime) {
    divclass = empty(divclass) ? 'ajax_form' : divclass;
    lefttime = empty(lefttime) ? 999999 : lefttime;
    xctrlid = ctrlid;
    aj_menu(e, ctrlid, divclass, lefttime);
}
function aj_menu(e, ctrlid, divclass, lefttime) {
    divclass = empty(divclass) ? 'ajax_msg' : divclass;
    lefttime = empty(lefttime) ? 2 : lefttime;
    var div = $(ctrlid + '_menu');
    if(empty(div)) {
        div = $c('div');
        div.id = ctrlid + '_menu';
        div.style.display = 'none';
        $('supevbox').appendChild(div);
    }
    div.className = divclass;
    var x = new ajax_cls();
    x.ctrlid = ctrlid;
    var href = isset($(x.ctrlid).href) ? $(x.ctrlid).href : $(x.ctrlid).
    attributes['href'].value;
    x.div = div;
    x.get(href + '&inajax=1&ajaxtarget='+x.ctrlid+'_menu',function(s, x) {
        _evalscript(s);
        s = _stripscript(s);
        if(x.div) {
            x.div.innerHTML = s;
        }
        showMenu(x.ctrlid, true, 0, lefttime);
    });
    doane(e);
}
```

在代码 18.18 中实现了分享班级的功能，由于代码比较简单，在此就不再赘述。运行程序，分享班级页面效果如图 18-6 所示。



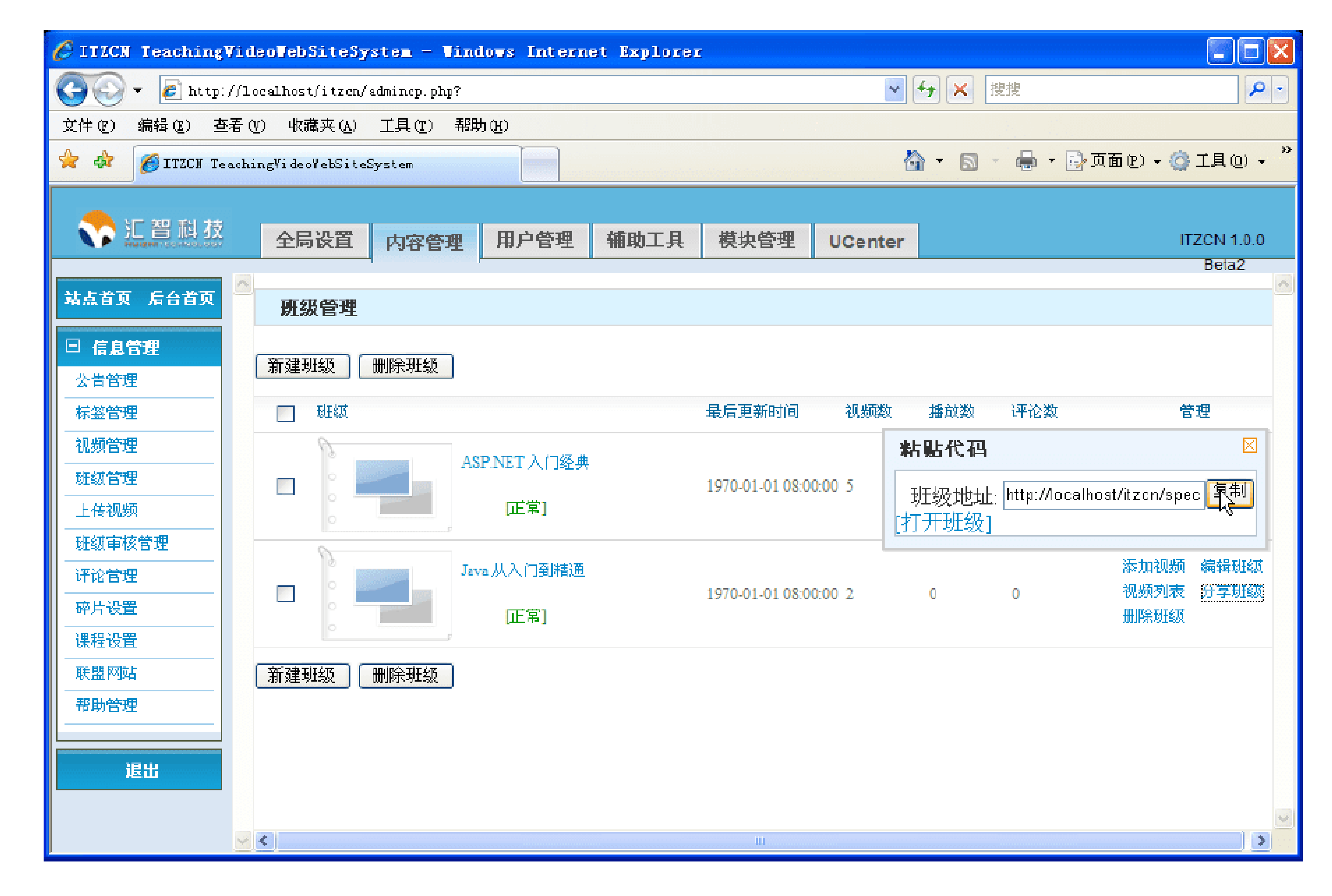


图 18-6　分享班级效果图

18.5.3　添加视频和视频列表

在上一小节中讲解了如何获取班级列表和删除班级等功能，本小节将讲解如何为班级组织课程表和管理视频列表。班级课程表其实就是向班级中添加视频信息，并且以章节的形式保存班级视频列表。下面首先讲解如何实现班级组织课程表。

在添加视频页面，首先运用 Ajax 技术异步获取课程中的视频信息，然后添加到班级课程表中。由于该页面设计比较简单，在此就不再讲解如何实现界面元素。获取视频信息首先需要选择课程，然后单击【搜索】按钮获取视频信息。单击【搜索】按钮会执行 JavaScript 脚本程序 searchvideo()函数，由于该函数比较简单，在此就不再赘述。执行 searchvideo()函数时会直接调用 aj_submit()函数，aj_submit()函数主要是获取课程中的视频。aj_submit()函数具体程序如代码 18.19 所示。

代码 18.19　获取视频信息程序

```
function aj_submit(formid, outid, insert, except, loading) {
    var x = new ajax_cls();
    var param = getform(formid) + '&inajax=1';
    insert = empty(insert) ? 0 : insert;
    except = empty(except) ? '' : except;
    loading = empty(loading) ? false : true;
    loadingstr = '';
    x.outid = outid;
    if(loading == true) {
```

```
            loadingstr = "<p class='info'><img src='"+jsenv['THEME_PATH']
            +"res/loading.gif'>loading...</p>";
        }
        var tmpinnerHTML = $(x.outid).innerHTML;
        if(isset(except)) {
            if(tmpinnerHTML.indexOf(except) != -1) {
                tmpinnerHTML = '';
            }
        }
        if(loading == true) {
            if(insert != 0) {
                $(x.outid).innerHTML = loadingstr + tmpinnerHTML;
            } else {
                $(x.outid).innerHTML = loadingstr;
            }
        }
        x.post($(formid).action, param, function(s, x) {
                setMenuPosition($(x.outid).ctrlid, 0);
                setTimeout("hideMenu()", 3000);
                _evalscript(s);
                s = _stripscript(s);
                if(insert != 0) {
                    $(x.outid).innerHTML = s + tmpinnerHTML;
                } else {
                    $(x.outid).innerHTML = s;
                }
                _ajaxupdateevents($(x.outid));
        });
        return false;
    }
```

获取视频信息后就可以向课程表中添加视频。代码 18.19 会自动为每一个视频信息添加一个按钮，当用户单击该按钮时会自动向课程表中添加视频。如果该视频已经被添加过，系统会给出已经添加过的提示。接下来详细讲解如何保存视频列表信息，在课程表中可以修改视频的章名。如果没有章名，系统会给出提示。当用户修改章名时会调用 JavaScript 脚本函数，该函数如代码 18.20 所示。

代码 18.20　修改视频所在章名函数

```
    function editalias(vid) {
        var obj = $('alias' + vid);
        var tag = obj.firstChild.tagName;
        if(typeof(tag) != 'undefined' && tag.toLowerCase() == 'input') {
            return false;
        }
```

```
        var org = obj.innerHTML;
        var val = (browser.name == 'ie') ? obj.innerText : obj.textContent;
        var txt = document.createElement('INPUT');
        txt.value = (val == 'N/A') ? '' : val;
        txt.style.width = (obj.offsetWidth + 12) + 'px' ;
        obj.innerHTML = '';
        obj.appendChild(txt);
        txt.focus();
        txt.onkeydown = function(event) {
            if (event.keyCode == 13) {
                obj.blur();
                return false;
            }
            if (event.keyCode == 27) {
                obj.innerHTML = org;
            }
        }
        txt.onblur = function(event) {
            if (trim(txt.value).length > 0) {
                obj.innerHTML = trim(txt.value);
                videorecord[videoorder[vid]][2] = trim(txt.value);
            } else {
                obj.innerHTML = org;
            }
        }
        return false;
    }
```

在代码 18.20 中，函数的主要作用是修改原保存章名的数组，在保存班级时提交该数组。接下来讲解如何保存班级课程表，该表单提交的 action 值为 addvideo，因此当用户单击【保存班级】按钮时，PHP 程序会自动转到 action 值为 addvideo 的判断语句中执行，主要程序如代码 18.21 所示。

代码 18.21　保存班级处理程序

```
    foreach ( $vorder as $rvid => $rorder )
                        {
                                $svideoarr[$rvid]['spid'] = $spid;
                                $svideoarr[$rvid]['vid'] = $rvid;
                                $svideoarr[$rvid]['alias'] = empty
                                ($valias[$rvid]) ? $svideoarr[$rvid]['alias'] :
                                $valias[$rvid];
                                //===========新增===============
                                $svideoarr[$rvid]['zname'] = empty
                                ($vzname[$rvid])? $svideoarr[$rvid]['zname'] :
                                $vzname[$rvid];
```

```
                $svideoarr[$rvid]['displayorder'] = $rorder;
                $svideoarr[$rvid]['ctime'] =
                empty($svideoarr[$rvid]['ctime'])?$timestamp:
                $svideoarr[$rvid]['ctime'];
                $svideoarr[$rvid]['author'] = empty
                ( $svideoarr[$rvid]['author'] ) ?
                $nvideoarr[$rvid]['author'] :
                $svideoarr[$rvid]['author'];
                $svideoarr[$rvid]['authorid'] =
                empty( $svideoarr[$rvid]['authorid'] ) ?
                $nvideoarr[$rvid]['authorid'] :
                $svideoarr[$rvid]['authorid'];
                $svideoarr[$rvid]['ivid'] =
                empty( $svideoarr[$rvid]['ivid'] ) ?
                $nvideoarr[$rvid]['ivid'] :
                $svideoarr[$rvid]['ivid'];
                $svideoarr[$rvid]['thumb'] =
                empty( $svideoarr[$rvid]['thumb'] ) ?
                $nvideoarr[$rvid]['thumb'] :
                $svideoarr[$rvid]['thumb'];
        }
        if ( !empty( $svideoarr ) )
        {

                $sql = "REPLACE INTO {$tablepre}vspecials (spid,
                vid, alias, displayorder, ctime, utime,zname)
                VALUES";
                foreach ( $svideoarr as $svideo )
                {

                    if($svideo['zname']=='undefined')
                    {
                    $sql .= "('{$spid}', '{$svideo['vid']}',
                '{$svideo['alias']}',
                '{$svideo['displayorder']}',
                '{$svideo['ctime']}', '{$timestamp}',''),";
                    }
                    else
                    {
                    $sql .= "('{$spid}', '{$svideo['vid']}',
                '{$svideo['alias']}',
                '{$svideo['displayorder']}',
                '{$svideo['ctime']}',
                '{$timestamp}','{$svideo['zname']}'),";
                    }
                }

                $sql = substr( $sql, 0, 0 - 1 );

                $db->query( $sql );
        }
```

在代码 18.21 中，首先获取了界面上保存的原有章名和修改后的章名，以及视频的节名，最后将这些值保存到数据库中。此时组织课程列表页面就成功的完成了。运行程序，该页面效果如图 18-7 所示。



图 18-7　组织课程列表页面效果图

视频列表页面主要是根据班级显示班级课程表，以及管理这些视频信息，例如修改视频名称、删除班级课程表中的视频和排序功能。实现视频列表的方法和实现班级列表的方法基本相同，另外删除功能也和其相同，在此就不再对该页面详细讲解，只列出视频列表页面的效果图，页面效果如图 18-8 所示。

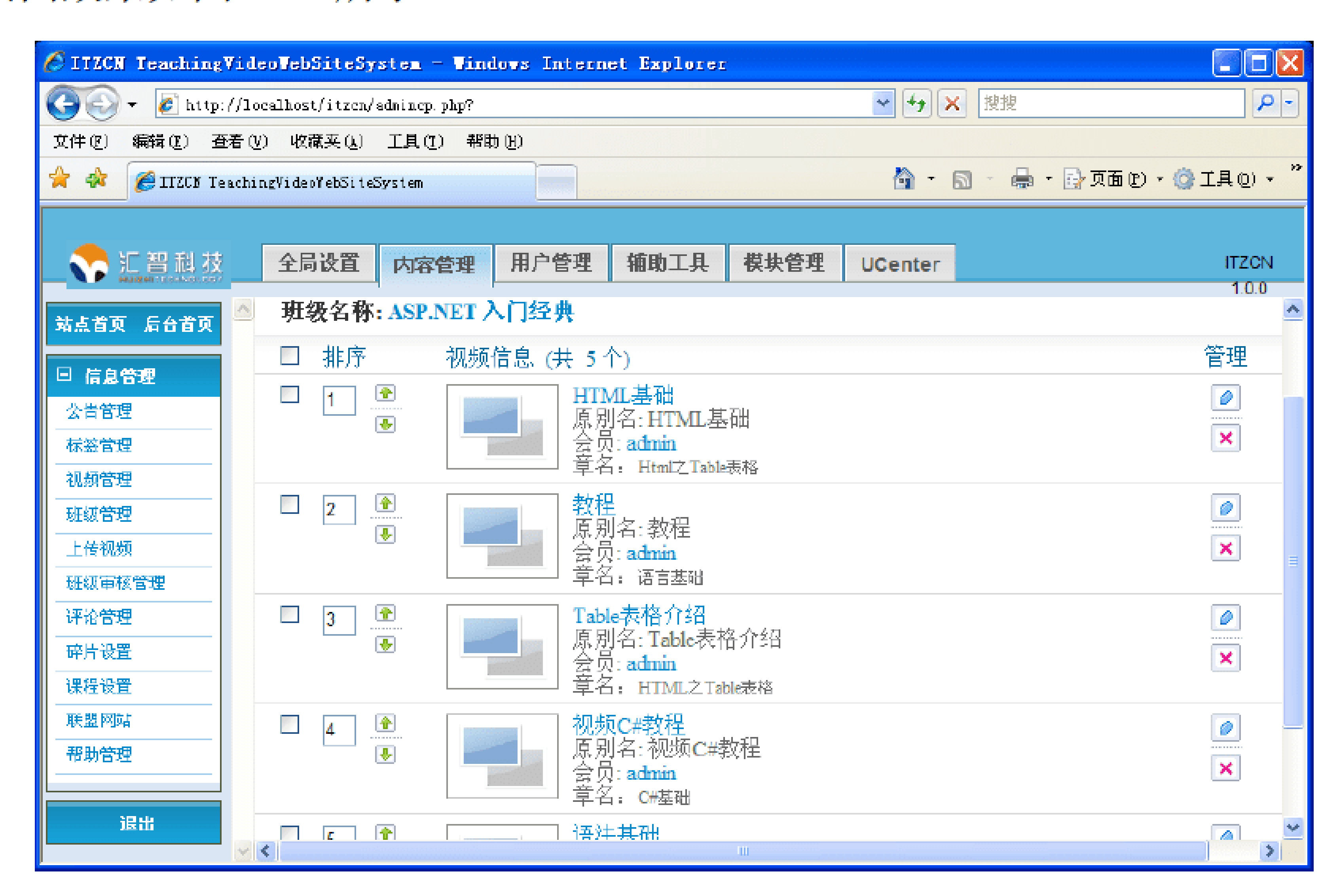


图 18-8　班级视频列表页面效果图